Evaluating the Security of Machine Learning Algorithms

Marco Antonio Barreno

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-63
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-63.html

May 20, 2008

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
20 MAY 2008		**00-00-2008 to 00-00-2008**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Evaluating the Security of Machine Learning Algorithms	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Electrical Engineering and Computer Sciences,Berkeley,CA,94720-1700**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**128**	

Evaluating the Security of Machine Learning Algorithms

by

Marco Antonio Barreno

A.B. (Dartmouth College) 2002

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor J. D. Tygar, Chair
Professor Anthony D. Joseph
Professor John Chuang

Spring 2008

The dissertation of Marco Antonio Barreno is approved:

Chair Date

Date

Date

University of California, Berkeley

Spring 2008

Evaluating the Security of Machine Learning Algorithms

Marco Antonio Barreno

Abstract

Evaluating the Security of Machine Learning Algorithms

by

Marco Antonio Barreno

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor J. D. Tygar, Chair

Two far-reaching trends in computing have grown in significance in recent years. First, statistical machine learning has entered the mainstream as a broadly useful tool set for building applications. Second, the need to protect systems against malicious adversaries continues to increase across computing applications. The growing intersection of these trends compels us to investigate how well machine learning performs under adversarial conditions. When a learning algorithm succeeds in adversarial conditions, it is an algorithm for *secure learning.* The crucial task is to evaluate the resilience of learning systems and determine whether they satisfy requirements for secure learning. In this thesis, we show that *the space of attacks against machine learning has a structure that we can use to build secure learning systems.*

This thesis makes three high-level contributions. First, we develop a framework

for analyzing attacks against machine learning systems. We present a taxonomy that describes the space of attacks against learning systems, and we model such attacks as a cost-sensitive game between the attacker and the defender. We survey attacks in the literature and describe them in terms of our taxonomy. Second, we develop two concrete attacks against a popular machine learning spam filter and present experimental results confirming their effectiveness. These attacks demonstrate that real systems using machine learning are vulnerable to compromise. Third, we explore defenses against attacks with both a high-level discussion of defenses within our taxonomy and a multi-level defense against attacks in the domain of virus detection. Using both global and local information, our virus defense successfully captures many viruses designed to evade detection. Our framework, exploration of attacks, and discussion of defenses provides a strong foundation for constructing secure learning systems.

Professor J. D. Tygar
Dissertation Committee Chair

For Jessica

Contents

List of Figures

List of Tables

Acknowledgments

First, I wish to thank my advisor, Doug Tygar, for the vital role he has played in my career as a graduate student. Doug has been a source of inspiration, encouragement, and guidance for me, and he has taught me much about how to do research well. All the success I have had at Berkeley is due to Doug's mentorship and support.

I would like to thank Professor Anthony Joseph for his guidance and advice. Through close collaboration he has helped me hone my research skills, and his feedback and suggestions on my work, especially this dissertation, have been invaluable.

I would like to thank Professor John Chuang for being on my qualifying exam and dissertation committees, and for the feedback he has given me on my research.

For useful feedback on my research and helpful discussions at various points, I thank Professor Michael Jordan, Professor Peter Bartlett, Professor Satish Rao, Professor Carla Brodley, Alvaro Cárdenas, Karl Chen, Byung-Gon Chun, Matt Harren, Chris Karlof, Adrian Mettler, and David Molnar.

For collaboration on the research culminating in this dissertation, I thank Jack Chi, Udam Saini, Charles Sutton, and Kai Xia. For particularly helpful collaboration in defining the early stages of my research, I thank Russell Sears. For especially helpful collaboration in advancing the later stages of my research, I thank Ben Rubinstein.

I would like to thank Blaine Nelson for all the ideas, hard work, and support that he has contributed to this research. Blaine has been an instrumental partner in developing the concepts and results presented here.

Finally, I wish to express my gratitude to my wife, Jessica Lemieux, for all of the patience with my crazy deadline schedules, feedback on my writing, care while I was busy, advice on hard decisions, and support in every way that she has given me during these years of graduate school. I am very lucky to have her.

Portions of the research presented in this thesis have appeared in previous publications [Barreno, Nelson, Sears, and Joseph, 2006a; Barreno, Nelson, Sears, Joseph, and Tygar, 2006b; Barreno, Nelson, Joseph, and Tygar, 2008; Nelson, Barreno, Chi, Joseph, Rubinstein, Saini, Sutton, Tygar, and Xia, 2008].

This research was supported in part by the Team for Research in Ubiquitous Secure Technology (TRUST), which receives support from the National Science Foundation (NSF award #CCF-0424422), the Air Force Office of Scientific Research (AFOSR #FA9550-06-1-0244), Cisco, British Telecom, ESCHER, Hewlett-Packard, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies; in part by California state Microelectronics Innovation and Computer Research Opportunities grants (MICRO ID#06-148 and #07-012) and Siemens; and in part by the cyber-DEfense Technology Experimental Research laboratory (DETERlab), which receives support from the Department of Homeland Security Homeland Security Advanced Research Projects Agency (HSARPA award #022412) and AFOSR (#FA9550-07-1-0501). The opinions expressed in this thesis are solely those of the author and do not necessarily reflect the opinions of any funding agency, the State of California, or the U.S. government.

Chapter 1

Introduction

1.1 Thesis

Two far-reaching trends in computing have grown in significance in recent years.

First, statistical machine learning has entered the mainstream as a broadly useful tool set for building applications. As a technique for building adaptive systems, machine learning enjoys several advantages over hand-crafted rules and other approaches: it can infer hidden patterns in data, it can adapt quickly to new signals and behaviors, and it can provide statistical soundness.

Second, the need to protect systems against malicious adversaries continues to increase across computing applications. Rising levels of hostile behavior have plagued application domains such as email, web search, and electronic commerce. Malicious activity has more recently become a problem for file sharing, instant messaging, mo-

bile phone applications, and other areas. Miscreants and criminals are becoming more organized in their attacks, and the need to secure applications against attackers is confronting all areas of computing.

The growing intersection of these trends compels us to investigate how well machine learning performs under adversarial conditions. Machine learning is useful because it can adapt to changing conditions, but attackers can influence those conditions to mislead the learner. Machine learning is powerful because it can extract complex hidden patterns from data, but attackers can obscure true patterns and inject their own. Machine learning selects the statistically best hypothesis, but attackers can reverse engineer the learner to find outliers and misclassifications.

When a learning algorithm succeeds in adversarial conditions, it is an algorithm for *secure learning*. The crucial task is to evaluate the resilience of learning systems and determine whether they satisfy requirements for secure learning.

We provide the foundation for this task. In this thesis, we show that *the space of attacks against machine learning has a structure that we can use to build secure learning systems.*

1.2 Contributions

Our task of making learning systems secure against attack comprises several steps. The first step is to discover what types of attack are possible, enumerating them and analyzing how they relate to each other. The second step is to understand the effects

of attacks and evaluate how dangerous they can be. The final step is to construct general approaches to protecting systems against such attacks.

This thesis provides the foundation for a comprehensive understanding of attacks against machine learning systems and strategies for defending against them. We address what attacks are possible by developing a taxonomy of attacks against machine learning systems, showing how it describes attacks in the literature and others. We explore the danger attacks pose by constructing successful attacks against the popular machine learning spam filter SpamBayes and measuring their effects. We begin to construct general approaches to protecting systems by examining each class within the taxonomy of attacks and discussing what defenses might be appropriate, and by developing a technique that combines specific and general information to detect new email viruses. Since spam filtering and virus/intrusion detection are two common and typical domains for using machine learning in an adversarial environment, these in-depth studies provide insight that is useful in general.

The importance of information for attackers and defenders is a pervasive theme throughout this work. In Chapter 3, we explore how the amount of information available to an attacker affects the success of attacks and even determines what attacks are possible. Information about the word distribution or exact contents of emails allows an attacker to construct more successful attacks on an email spam filter. In Chapter 4, we examine two different levels of information in email virus detection: global (cross-user) and local (user-specific). We investigate combining the two types

of information with a *Latent Dirichlet Allocation (LDA)* model to generalize quickly while also learning each user's specific behavior patterns. Finally, we discuss several open problems related to information in Chapter 5.

This thesis makes the following specific contributions:

- *A framework for analyzing attacks against machine learning systems.*

  We propose a taxonomy of attacks against machine learning systems that categorizes attacks based on key distinguishing characteristics. In particular, we show that there are at least three interesting dimensions to potential attacks against learning systems: (1) they may be *Causative* in their influence over the training process, or they may be *Exploratory* and take place post-training; (2) they may be attacks on *Integrity* aimed at *false negatives* (allowing hostile input into a system) or they may be attacks on *Availability* aimed at *false positives* (preventing benign input from entering a system); and (3) they may be *Targeted* at a particular input or they may be *Indiscriminate* in which inputs fail. Each of these dimensions operates independently, so we have at least eight distinct classes of attacks on machine learning systems.

  We model learning in an adversarial environment as a cost-sensitive game between an *attacker* and a *defender*; the taxonomy determines both the structure of the game and the type of cost model. We present the taxonomy and learning game in Section 2.2.

- *A survey of attacks in the literature within the framework.*

  In recent years, many researchers have explored attacks against machine learning systems and defenses to make those learners more secure. In Section 2.3, we survey the literature and describe attacks in terms of our taxonomy, finding connections between different approaches. We highlight which types of attack have received substantial attention in the research community and which areas have not been as fully explored.

- *A discussion of potential defensive techniques organized by our framework.*

  The purpose of this work is to build a foundation for secure learning. In Section 2.4 we discuss general approaches for defending against attacks in each area of the attack space defined by our taxonomy. Several defensive techniques have appeared in the literature, and we present these techniques alongside new ideas for making learning systems secure against attack. We propose defense ideas and highlight promising areas for further research.

- *Two novel attacks against a machine learning spam filter.*

  We develop two novel attacks against the SpamBayes statistical spam filter. Using the taxonomy to identify regions of attacks space that have not received much prior attention, we explore attacks that manipulate training data to cause certain forms of *denial of service*. We describe our *dictionary attack* and *focused attack* in Section 3.3.

- *Experiments demonstrating the effectiveness of our attacks.*

  We present experimental results in Section 3.4 confirming that these attacks present a serious concern for statistical spam filters. A *dictionary attack* can make a spam filter unusable by controlling just 1% of the messages in the training set. A *focused attack* can successfully prevent victims from receiving specific email messages 90% of the time.

  These attacks demonstrate the vulnerability of real learning systems, and they serve as an example of how our framework can guide and analyze attacks.

- *A multi-level defense against email virus attacks.*

  The use of high-level (*global*) information vs. focused (*local*) information is in some sense the defender's analog of *Indiscriminate* or *Targeted* attack motive. In Section 4.2, we adapt the LDA model to the domain of virus detection in a population of email users, making use of both global (cross-user) and local (user-specific) information to defend against attacks.

- *Experiments evaluating the success of our defense.*

  We present experimental results evaluating our virus defense in Section 4.3, using traces generated by real viruses. Our experiments show that a combination of global and local information can be beneficial when detecting viruses.

In Chapter 2 we present our framework for attacks against machine learning and show how it organizes attacks in the literature and suggests defensive strategies.

In Chapter 3 we pursue two attacks against the SpamBayes statistical spam filter, demonstrating their effectiveness with strong experimental results. We explore a defensive technique that combines high-level and low-level information to identify email virus attacks in Chapter 4. Finally, in Chapter 5 we discuss open problems in this field and give some concluding thoughts.

Chapter 2

A Framework for Secure Learning

In this chapter, we introduce our framework for analyzing attacks on machine learning systems.[1] We introduce the learning problem in an adversarial setting and we detail the threat model. We present our taxonomy, which categorizes attacks against machine learning systems along three axes, and we show that attacks on learning systems have a clean game structure. We review attacks in the literature in the context of our taxonomy, and we discuss defensive strategies.

2.1 Notation and setup

We focus on binary classification for security applications, in which a *defender* attempts to separate *instances* of input (data points), some or all of which come

[1] Material in this chapter has been previously published in the papers "Can machine learning be secure?" [Barreno et al., 2006b] and "The security of machine learning" [Barreno et al., 2008].

from a malicious *attacker*, into harmful and benign classes. This setting covers many interesting security applications, such as host and network intrusion detection, virus and worm detection, and spam filtering. In detecting malicious activity, the *positive* class (label 1) indicates malicious *intrusion* instances while the *negative* class (label 0) indicates benign *normal* instances. A classification error is a *false positive (FP)* if a normal instance is classified as positive (malicious) and a *false negative (FN)* if an intrusion instance is classified as negative (benign).

In the *supervised classification* problem, the learner trains on a *training set* of N instances, $\mathbf{X} = \{(\mathbf{x}, y)|\mathbf{x} \in \mathcal{X}, y \in \mathcal{Y}\}^N$, given an instance space $\mathcal{X}$ and the label space $\mathcal{Y} = \{0, 1\}$, and we evaluate its performance on an *evaluation set* $\mathbf{E}$ for which the labels are unknown until after classification. Given some hypothesis class Ω, the goal is to learn a classification hypothesis (classifier) $f^* \in \Omega$ to minimize errors when predicting labels for new data, or if our model includes a cost function over errors, to minimize the total cost of errors. The cost function C assigns a numeric cost to each combination of data instance, true label, and classifier label. The defender chooses a *learning algorithm*, or *procedure*, D for selecting hypotheses. The classifier may periodically interleave *training* steps with the *evaluation*, retraining on some or all of the accumulated old and new data. In adversarial environments, the attacker controls some of the data, which may be used for training.

We show the training setup and flow of information in Figure 2.1.

The procedure can be any method of selecting a hypothesis; in statistical machine

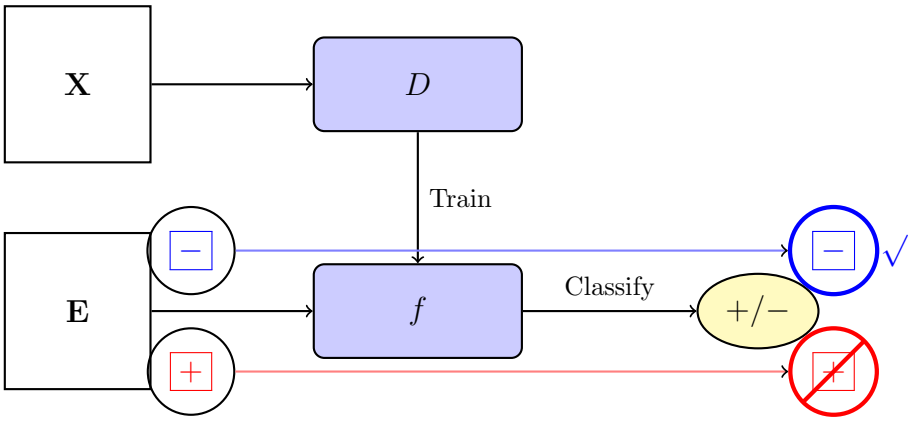


Figure 2.1: The basic training and classification setup. The learning algorithm D trains on training data $\mathbf{X}$ and produces classifier f. The classifier provides classifications for the evaluation data $\mathbf{E}$: it accepts normal points ($-$) and rejects intrusion points ($+$). When the learner retrains, it merges $\mathbf{E}$ into $\mathbf{X}$ and produces a new classifier f to classify future evaluation data.

learning, the most common type of procedure is *(regularized) empirical risk minimization*. This procedure is an optimization problem where the objective function has an *empirical risk* term and a *regularization* term. Since true cost is often not representable precisely and efficiently, we calculate risk as the expected *loss* given by a *loss function* ℓ that approximates true cost; the regularization term ρ captures some notion of hypothesis complexity to prevent *overfitting* the training data. This procedure finds the hypothesis minimizing:

$$f^* = \operatorname*{argmin}_{f \in \Omega} \sum_{(\mathbf{x},y) \in \mathbf{X}} \ell(y, f(\mathbf{x})) \quad + \quad \lambda \rho(f) \tag{2.1}$$

Many learning methods make a *stationarity* assumption: training data and evaluation data are drawn from the same distribution. This assumption allows us to minimize the risk on the training set as a surrogate for risk on the evaluation data, since evaluation data are not known at training time. However, real-world sources

of data often are not stationary and, even worse, attackers can easily violate the stationarity assumption with some control of either training or evaluation instances. Analyzing and strengthening learning methods in the face of a broken stationarity assumption is the crux of the *secure learning* problem.

We model attacks on machine learning systems as a game between two players, the *attacker* and the *defender*. The game consists of a series of *moves*, or *steps*. Each move encapsulates a choice by one of the players: the attacker alters or selects data; the defender chooses a training procedure for selecting the classification hypothesis.

A summary of the notation we use in this thesis, along with a glossary of terms, appears after the text.

2.2 Framework

2.2.1 Security analysis

Our framework proceeds from careful security analysis of the machine learning setting. Properly analyzing the security of a system requires identifying *security goals* and a *threat model*. Security is concerned with protecting assets from attackers. A security goal is a requirement that, if violated, results in the partial or total compromise of an asset. A threat model is a profile of attackers, describing motivation and capabilities. Here we analyze the security goals and threat model for machine learning systems.

We assume the use of a classifier for a security goal. For example, a virus detection system has the goal of preventing virus infection, and an intrusion detection system has the goal of preventing malicious intrusion. A virus infection or a successful intrusion has high cost relative to other outcomes. In this section we describe security goals and a threat model that are specific to machine learning systems.

Security goals

In a security context the classifier's purpose is to classify malicious events and prevent them from interfering with system operations. We split this general learning goal into two goals:

- *Integrity goal:* To prevent attackers from reaching system assets.
- *Availability goal:* To prevent attackers from interfering with normal operation.

There is a clear connection between false negatives and violation of the integrity goal: malicious instances that pass through the classifier can wreak havoc. Likewise, false positives tend to violate the availability goal because the learner itself denies benign instances.

Threat model

The threat model for machine learning comprises the attacker's goals/incentives and capabilities.

Attacker goal/incentives. In general the attacker wants to access system assets (with false negatives) or deny normal operation (usually with false positives). For example, a virus author wants viruses to pass through the filter and take control of the protected system (a false negative). On the other hand, an unscrupulous merchant may want sales traffic to a competitor's web store to be blocked as intrusions (false positives).

The attacker may have a short-term goal, such as to find a single false negative, or the attacker may have a long-term goal, such as to consistently find false negatives into the future.

We assume that the attacker and defender each have a *cost function* that assigns a cost to each labeling for any given instance. Cost can be positive or negative; a negative cost is a benefit. It is usually the case that low cost for the attacker parallels high cost for the defender and vice-versa; the attacker and defender would not be adversaries if their goals aligned. An important special case is a *zero-sum game*, in which the sum of the attacker's cost and the defender's cost is zero (or any other fixed value) for each possible outcome. Zero-sum games are particularly amenable to analysis, and they can be good approximations for true cost functions (which are rarely exactly zero-sum). In this thesis, we generally assume that games are zero-sum. We take the defender's point of view, so "high-cost" means high positive cost for the defender.

Attacker capabilities. We assume that the attacker has knowledge of the training algorithm, and in many cases partial or complete information about the training set, such as its distribution. The attacker may be able to modify or generate data used in training; we consider cases in which the attacker can and cannot control some of the learner's training data.

In general we assume the attacker can generate arbitrary instances; however, many specific problems impose reasonable restrictions on the attacker's ability to generate instances. For example, when the learner trains on data from the attacker, sometimes it is safe to assume that the attacker cannot choose the label for training. As another example, an attacker may have complete control over data packets being sent from the attack source, but routers in transit may add to or alter the packets as well as affect their timing and arrival order.

When the attacker controls training data, an important limitation to consider is what fraction of the training data the attacker can control and to what extent. If the attacker has arbitrary control over 100% of the training data, it is difficult to see how the learner can learn anything useful; however, even in such cases there are learning strategies that can make the attacker's task more difficult (see Section 2.4.2). We primarily examine intermediate cases and explore how much influence is required for the attacker to defeat the learning procedure.

2.2.2 Taxonomy

We present a taxonomy categorizing attacks on learning systems along three axes:

Influence

- *Causative* attacks influence learning with control over training data.
- *Exploratory* attacks exploit misclassifications but do not affect training.

Security violation

- *Integrity* attacks compromise assets via false negatives.
- *Availability* attacks cause denial of service, usually via false positives.

Specificity

- *Targeted* attacks focus on a particular instance.
- *Indiscriminate* attacks encompass a wide class of instances.

The first axis describes the capability of the attacker: whether (a) the attacker has the ability to influence the training data that is used to construct the classifier (a *Causative* attack) or (b) the attacker does not influence the learned classifier, but can send new instances to the classifier and possibly observe its decisions on these carefully crafted instances (an *Exploratory* attack). In one sense, *Causative* attacks are more fundamentally learning attacks than *Exploratory* attacks are: while many types of systems perform poorly on cleverly modified instances, only systems that

learn from data can be misled by an attacker to form incorrect models, choosing poor hypotheses. On the other hand, the hypotheses produced by learning algorithms have certain regularities and structures that an attacker may be able to exploit in an *Exploratory* attack, so it is certainly worthwhile to consider them carefully alongside *Causative* attacks.

The second axis indicates the type of security violation the attacker causes: (a) to create false negatives, in which harmful instances slip through the filter (an *Integrity* violation); or (b) to create a denial of service, usually by inducing false positives, in which benign instances are incorrectly filtered (an *Availability* violation).

The third axis refers to how specific the attacker's intention is: whether (a) the attack is highly *Targeted* to degrade the classifier's performance on one particular instance or (b) the attack aims to cause the classifier to fail in an *Indiscriminate* fashion on a broad class of instances. Each axis, especially this one, is actually a spectrum of choices.

The **Influence** axis of the taxonomy determines the structure of the game and the move sequence. The **Specificity** and **Security violation** axes of the taxonomy determine the general shape of the cost function: an *Integrity* attack benefits the attacker on false negatives, and therefore focuses high cost (to the defender) on false negatives, and an *Availability* attack focuses high cost on false positives; a *Targeted* attack focuses high cost only on a small class of instances, while an *Indiscriminate* attack spreads high cost over a broad range of instances.

2.2.3 Examples

Here we give four hypothetical attack scenarios, each with two variants, against a machine learning intrusion detection system (IDS). Table 2.1 shows where these examples fit within the taxonomy. This section gives the reader an intuition for how the taxonomy organizes attacks against machine learning systems; Section 2.3 presents a similar table categorizing attacks published in the literature.

***Causative Integrity* attack: The intrusion foretold**

In a *Causative Integrity* attack, the attacker uses control over training to cause intrusions to slip past the classifier as false negatives.

Example: an attacker wants the defender's IDS not to block a novel virus. The defender trains periodically on network traffic, so the attacker sends non-intrusion traffic that is carefully chosen to look like the virus and mis-train the learner so that it fails to block it.

This example might be *Targeted* if the attacker already has a particular virus executable to send and needs to cause the learner to miss that particular instance. It might be *Indiscriminate*, on the other hand, if the attacker has a certain payload but could use any of a large number of existing exploit mechanisms to transmit the payload, in which case the attack need only fool the learner on any one of the usable executables.

	Integrity	*Availability*
Causative: *Targeted*	*The intrusion foretold*: mis-train a particular intrusion	*The rogue IDS*: mis-train IDS to block certain traffic
Indiscriminate	*The intrusion foretold*: mis-train any of several intrusions	*The rogue IDS*: mis-train IDS to broadly block traffic
Exploratory: *Targeted*	*The shifty intruder*: obfuscate a chosen intrusion	*The mistaken identity*: censor a particular host
Indiscriminate	*The shifty intruder*: obfuscate any intrusion	*The mistaken identity*: interfere with traffic generally

Table 2.1: Our taxonomy of attacks against machine learning systems, with examples from Section 2.2.3.

Causative Availability attack: The rogue IDS

In a *Causative Availability* attack, the attacker uses control over training instances to interfere with operation of the system, such as by blocking legitimate traffic.

Example: an attacker wants traffic to be blocked so the destination doesn't receive it. The attacker generates attack traffic similar to benign traffic when the defender is collecting training data to train the IDS. When the learner re-trains on the attack data, the IDS will start to filter away benign instances as if they were intrusions.

This attack could be *Targeted* at a particular protocol or destination. On the other hand, it might be *Indiscriminate* and attempt to block a significant portion of all legitimate traffic.

Exploratory Integrity attack: The shifty intruder

In an *Exploratory Integrity* attack, the attacker crafts intrusions so as to evade the classifier without direct influence over the classifier itself.

Example: an attacker modifies and obfuscates intrusions, such as by changing network headers and reordering or encrypting contents. If successful, these modifications prevent the IDS from recognizing the altered intrusions as malicious, so it allows them into the system.

In the *Targeted* version of this attack, the attacker has a particular intrusion to get past the filter. In the *Indiscriminate* version, the attacker has no particular preference and can search for any intrusion that succeeds, such as by modifying a large number

of different exploits to see which modifications evade the filter.

***Exploratory Availability* attack: The mistaken identity**

In an *Exploratory Availability* attack, the attacker interferes with normal operation without influence over training.

Example: an attacker sends intrusions appearing to originate from an innocent machine. The IDS, which has learned to recognize intrusions, blocks that machine.

In the *Targeted* version, the attacker has a particular machine to target. In the *Indiscriminate* version, the attacker may select any convenient machine or may switch IP addresses among many machines to induce greater disruption.

2.2.4 The adversarial learning game

This section models attacks on learning systems as games where moves represent strategic choices. The choices and computations in a move depend on information from previous moves (when a game is repeated, this includes previous iterations).

Exploratory game

We first present the game for *Exploratory* attacks:

1. **Defender** Choose procedure D for selecting hypothesis
2. **Attacker** Choose procedure A_E for selecting distribution
3. Evaluation:

- Reveal distribution $\mathbb{P}_{\mathrm{T}}$
- Sample dataset $\mathbf{X}$ from $\mathbb{P}_{\mathrm{T}}$
- Compute $f \leftarrow D(\mathbf{X})$
- Compute $\mathbb{P}_{\mathrm{E}} \leftarrow A_E(\mathbf{X}, f)$
- Sample dataset $\mathbf{E}$ from $\mathbb{P}_{\mathrm{E}}$
- Assess total cost: $\sum_{(\mathbf{x},y)\in\mathbf{E}} C\left(\mathbf{x}, f(\mathbf{x}), y\right)$

The defender's move is to choose a learning algorithm (procedure) D for creating hypotheses from datasets. For example, the defender may choose a *support vector machine* (SVM) with a particular kernel, loss, regularization, and cross-validation plan.

The attacker's move is then to choose a procedure A_E to produce a distribution on which to evaluate the hypothesis that D generates. (The degree of control the attacker has in generating the dataset is setting-specific.)

After the defender and attacker have both made their choices, the game is evaluated. A training dataset $\mathbf{X}$ is drawn from some fixed and possibly unknown distribution $\mathbb{P}_{\mathrm{T}}$, and training produces $f = D(\mathbf{X})$. The attacker's procedure A_E produces distribution $\mathbb{P}_{\mathrm{E}}$, which may be based on $\mathbf{X}$ and f, and an evaluation dataset $\mathbf{E}$ is drawn from $\mathbb{P}_{\mathrm{E}}$. Finally, the attacker and defender incur cost based on the performance of f evaluated on $\mathbf{E}$.

In many cases, the procedure A_E can *query* the classifier, treating it as an oracle that provides labels for query instances. Attacks that use this technique are *probing attacks*. Probing can reveal information about the classifier. On the other hand, with sufficient prior knowledge about the training data and algorithm, the attacker may be able to find high-cost instances without probing.

Causative game

The game for *Causative* attacks is similar:

1. **Defender** Choose procedure D for selecting hypothesis
2. **Attacker** Choose procedures A_T and A_E for selecting distributions
3. Evaluation:
   - Compute $\mathbb{P}_{\mathrm{T}} \leftarrow A_T$
   - Sample dataset $\mathbf{X}$ from $\mathbb{P}_{\mathrm{T}}$
   - Compute $f \leftarrow D(\mathbf{X})$
   - Compute $\mathbb{P}_{\mathrm{E}} \leftarrow A_E(\mathbf{X}, f)$
   - Sample dataset $\mathbf{E}$ from $\mathbb{P}_{\mathrm{E}}$
   - Assess total cost: $\sum_{(\mathbf{x},y)\in\mathbf{E}} C\left(\mathbf{x}, f(\mathbf{x}), y\right)$

This game is very similar to the *Exploratory* game, but the attacker can choose A_T to affect the training data $\mathbf{X}$. The attacker may have various types of influence over

the data, ranging from arbitrary control over some fraction of instances to a small biasing influence on some aspect of data production; details depend on the setting.

Control over training data opens up new strategies to the attacker. Cost is based on the interaction of f and $\mathbf{E}$. In the *Exploratory* game the attacker chooses $\mathbf{E}$ while the defender controls f; in the *Causative* game the attacker also influences f. With this influence, the attacker can proactively cause the learner to produce bad classifiers.

Iteration

We have described these games as *one-shot games*, in which players minimize cost when each move happens only once. We can also consider an *iterated game*, in which the game repeats several times and players minimize total accumulated cost. In this setting, we assume players have access to all information from previous iterations of the game. One-shot games naturally fit with short-term attacker goals, while iterated games can more easily accommodate attackers with long-term goals.

2.3 Attacks: Categorizing related work

This section surveys examples of learning in adversarial environments from the literature. Our taxonomy provides a basis for evaluating the resilience of the systems described, analyzing the attacks against them in preparation for constructing defenses. We organize attacks in the literature according to our taxonomy in this section and display them in Table 2.2.

	Integrity	*Availability*
Causative: *Targeted*	Kearns and Li [1993], Newsome, Karp, and Song [2006]	Kearns and Li [1993], Newsome et al. [2006], Chung and Mok [2007], Nelson et al. [2008]
Indiscriminate	Kearns and Li [1993], Newsome et al. [2006]	Kearns and Li [1993], Newsome et al. [2006], Chung and Mok [2007], Nelson et al. [2008]
Exploratory: *Targeted*	Tan, Killourhy, and Maxion [2002], Lowd and Meek [2005a], Wittel and Wu [2004], Lowd and Meek [2005b], Karlberger, Bayler, Kruegel, and Kirda [2007]	Moore, Shannon, Brown, Voelker, and Savage [2006]
Indiscriminate	Fogla and Lee [2006], Lowd and Meek [2005a], Wittel and Wu [2004], Karlberger et al. [2007]	Moore et al. [2006]

Table 2.2: Related work in the taxonomy.

2.3.1 *Causative Integrity* attacks

Contamination in PAC learning. Kearns and Li [1993] extend Valiant's *probably approximately correct* (PAC) learning framework [Valiant, 1984, 1985] to prove bounds for maliciously chosen errors in the training data. In PAC learning, an algorithm succeeds if it can, with probability at least $1 - \delta$, learn a hypothesis that has at most probability ϵ of making an incorrect prediction on an example drawn from the same distribution. Kearns and Li examine the case where an attacker has arbitrary control over some fraction of the training examples. They prove that in general the attacker can prevent the learner from succeeding if the fraction is at least $\epsilon/(1 + \epsilon)$, and for some classes of learners they show this bound is tight.

This work provides an interesting and useful bound on the ability to succeed at PAC-learning. The analysis broadly concerns both *Integrity* and *Availability* attacks as well as both *Targeted* and *Indiscriminate*. However, not all learning systems fall into the PAC-learning model.

Red herring attack. Newsome, Karp, and Song [2006] present *Causative Integrity* and *Causative Availability* attacks against Polygraph [Newsome, Karp, and Song, 2005], a polymorphic virus detector that learns virus signatures using both a conjunction learner and a naive-Bayes-like learner. Their *red herring* attacks against conjunction learners exploit certain weaknesses not present in other learning algorithms (these are *Causative Integrity* attacks, both *Targeted* and *Indiscriminate*).

2.3.2 *Causative Availability* attacks

Correlated outlier attack. Newsome et al. [2006] also suggest a *correlated outlier* attack, which attacks a naive-Bayes-like learner by adding spurious features to positive training instances, causing the filter to block benign traffic with those features (an *Availability* attack).

Allergy attack. Chung and Mok [2006, 2007] present *Causative Availability* attacks against the Autograph worm signature generation system [Kim and Karp, 2004]. Autograph operates in two phases. First, it identifies infected nodes based on behavioral patterns, in particular scanning behavior. Second, it observes traffic from the identified nodes and infers blocking rules based on observed patterns. Chung and Mok describe an attack that targets traffic to a particular resource. In the first phase, an attack node convinces Autograph that it is infected by scanning the network. In the second phase, the attack node sends crafted packets mimicking targeted traffic, causing Autograph to learn rules that block legitimate access and create a denial of service.

Attacking SpamBayes. We have developed *Causative Availability* attacks (both *Targeted* and *Indiscriminate*) against the SpamBayes statistical spam classifier [Nelson et al., 2008]. Our *dictionary attack* and *focused attack* can render SpamBayes unusable or prevent the victim from receiving a particular email message with only a small amount of influence on training. We present this work in Chapter 3.

2.3.3 *Exploratory Integrity* attacks

Some *Exploratory Integrity* attacks mimic statistical properties of the normal traffic to camouflage intrusions. In the *Exploratory* game, the attacker's move produces instances $\mathbf{E}$ that statistically resemble normal traffic in the training data $\mathbf{X}$ as measured by the learning procedure D.

Polymorphic blending attack. *Polymorphic blending attacks* encrypt attack traffic in such a way that it appears statistically identical to normal traffic. Fogla and Lee [2006] present a framework for reasoning about and generating *polymorphic blending attack* instances to evade intrusion detection systems.

Attacking `stide`. Tan, Killourhy, and Maxion [2002] describe a mimicry attack against the `stide` anomaly-based intrusion detection system (IDS). They modify exploits of the `passwd` and `traceroute` programs to accomplish the same end results using different sequences of system calls: the shortest subsequence in attack traffic that does not appear in normal traffic is longer than the IDS window size, so the attacker evades detection.

Good word attacks. Several authors demonstrate *Exploratory Integrity* attacks using similar principles against spam filters. Lowd and Meek [2005a] and Wittel and Wu [2004] develop attacks against statistical spam filters that add *good words*, or words the filter considers indicative of non-spam, to spam emails. Karlberger,

Bayler, Kruegel, and Kirda [2007] study the effect of replacing strong spam words with synonyms. These modification can make spam emails appear innocuous to the filter, especially if the words are chosen to be ones that appear often in non-spam email and rarely in spam email.

Reverse engineering classifiers. In another paper, Lowd and Meek [2005b] approach the *Exploratory Integrity* attack problem from a different angle: they give an algorithm for an attacker to reverse engineer a classifier. The attacker seeks the highest cost (lowest cost for the attacker) instance that the classifier labels *negative*. This work is interesting in its use of a cost function over instances for the attacker rather than simple positive/negative classification. We explore this work in more detail in Section 2.4.1.

2.3.4 *Exploratory Availability* attacks

Exploratory Availability attacks against non-learning systems abound in the literature: almost any *denial of service* (DoS) attack falls into this category, such as those described by Moore, Shannon, Brown, Voelker, and Savage [2006].

However, *Exploratory Availability* attacks against the learning components of systems are not common. We describe one possibility in Section 2.2.3: if a learning IDS has trained on intrusion traffic and has the policy of blocking hosts that originate intrusions, an attacker could send intrusions that appear to originate from a legitimate

host, causing the IDS to block that host. Another possibility is to take advantage of a computationally expensive learning component: for example, spam filters that use image processing to detect advertisements in graphical attachments can take significantly more time than text-based filtering [Dredze, Gevaryahu, and Elias-Bachrach, 2007; Wang, Josephson, Lv, Charikar, and Li, 2007]. An attacker could exploit such overhead by sending many emails with images, causing the expensive processing to delay and perhaps even block messages.

2.4 Defenses: Applying our framework

We discuss several defense strategies against broad classes of attacks. The game between attacker and defender and the taxonomy that we introduce in Section 2.3 provides a foundation on which to construct defenses. We address *Exploratory* and *Causative* attacks separately, and we also discuss the broader setting of an iterated game. In all cases, we must expect a trade-off: changing the algorithms to make them more robust against (worst-case) attacks will often make them *less* effective on average.

2.4.1 Defending against *Exploratory* attacks

Exploratory attacks do not corrupt the training data but attempt to find vulnerabilities in the learned hypothesis. The attacker attempts to construct a distribution

that concentrates probability mass on high-cost instances; in other words, the attacker tries to find an evaluation distribution on which the learner predicts poorly (violating stationarity). This section examines defender strategies that make it difficult for the attacker to construct such a distribution.

In the *Exploratory* game, the defender makes a move before observing contaminated data. The defender can impede the attacker's ability to reverse engineer the classifier by limiting access to information about the training procedure and data. With less information, the attacker has difficulty producing a distribution unfavorable to the defender. Nonetheless, even with incomplete information, the attacker may be able to construct an unfavorable evaluation distribution using a combination of prior knowledge and probing.

Defenses against attacks without probing

Part of our security analysis involves identifying aspects of the system that should be kept secret. In securing a learner, we attempt to limit information to make it difficult for an attacker to conduct an attack.

Training data. Keeping training data secret from the attacker limits the attacker's ability to reconstruct internal states of the classifier. There is a tension between collecting training data that fairly represents the real world instances and keeping all aspects of that data secret. In most situations, it is difficult to use completely secret training data, though the attacker may have only partial information about it.

Feature selection. We can make classifiers hard to reverse engineer through feature selection. Feature selection is the process of choosing a feature map that maps raw measurements into a new feature space on which a hypothesis is selected. Keeping secret which features are selected in learning, or choosing a secret mapping to a different feature space entirely, may hinder an attacker in finding high-cost instances.

Globerson and Roweis [2006] present a defense for the *Exploratory* attack of *feature deletion* on the evaluation data: features present in the training data, and perhaps highly predictive of an instance's class, are removed from the evaluation data by the attacker. For example, words present in training emails may not occur in evaluation messages, and network packets in training data may contain values for optional fields that are missing from future traffic. Globerson and Roweis formulate a modified support vector machine classifier robust against deletion of high-value features.

Obfuscation of spam-indicating words (an attack on the feature set) is a common *Targeted Exploratory Integrity* attack. Sculley, Wachman, and Brodley [2006] use inexact string matching to defeat obfuscations of words in spam emails. They use features based on character subsequences that are robust to character addition, deletion, and substitution.

Hypothesis space/learning procedures. A complex hypothesis space may make it difficult for the attacker to infer precise information about the learned hypothesis. However, hypothesis complexity must be balanced with capacity to generalize, such as through regularization.

Wang, Parekh, and Stolfo [2006] present *Anagram*, an anomaly detection system using *n-gram* models of bytes to detect intrusions. They incorporate two techniques to defeat *Exploratory* attacks that mimic normal traffic (*mimicry attacks*): (1) they use high-order n-grams (with n typically between 3 and 7), which capture differences in intrusion traffic even when that traffic has been crafted to mimic normal traffic on the single-byte level; and (2) they randomize feature selection by randomly choosing several (possibly overlapping) subsequences of bytes in the packet and testing them separately, so the attack will fail unless the attacker ensures that not only the whole packet, but also any subsequence, mimics normal traffic.

Dalvi, Domingos, Mausam, Sanghai, and Verma [2004] develop a cost-sensitive game-theoretic classification defense to counter *Exploratory Integrity* attacks. In their model, the attacker can alter instance features but incurs a known cost for each change. The defender can measure each feature at a different known cost. Each has a known cost function over classification/true label pairs (not zero-sum). The classifier is a cost-sensitive naive Bayes learner that classifies instances to minimize its expected cost, while the attacker modifies features to minimize its own expected cost. Their defense constructs an adversary-aware classifier by altering the likelihood function of the learner to anticipate the attacker's changes. They adjust the likelihood that an instance is malicious by considering that the observed instance may be the result of an attacker's optimal transformation of another instance. This defense relies on two assumptions: (1) the defender's strategy is a step ahead of the attacker's strategy,

and (2) the attacker plays optimally against the original cost-sensitive classifier. It is worth noting that while their approach defends against optimal attacks, it doesn't account for non-optimal attacks. For example, if the attacker doesn't modify any data, the adversary-aware classifier misclassifies some instances that the original classifier correctly classifies.

Defenses against probing attacks

The ability to query a classifier gives an attacker additional attack options.

Analysis of reverse engineering. Lowd and Meek [2005b] observe that the attacker need not model the classifier explicitly, but only find lowest-attacker-cost instances as in the Dalvi et al. setting. They formalize a notion of reverse engineering as the *adversarial classifier reverse engineering* (ACRE) problem. Given an attacker cost function, they analyze the complexity of finding a lowest-attacker-cost instance that the classifier labels as negative. They assume no general knowledge of training data, though the attacker does know the feature space and also must have one positive example and one negative example. A classifier is *ACRE-learnable* if there exists a polynomial-query algorithm that finds a lowest-attacker-cost negative instance. They show that linear classifiers are ACRE-learnable with linear attacker cost functions and some other minor restrictions.

The ACRE-learning problem provides a means of qualifying how difficult it is to use queries to reverse engineer a classifier from a particular hypothesis class using a

particular feature space. We now suggest defense techniques that can increase the difficulty of reverse engineering a learner.

Randomization. A randomized hypothesis may decrease the value of feedback to an attacker. Instead of choosing a hypothesis $f : \mathcal{X} \to \{0, 1\}$, we generalize to hypotheses that predict a real value on $[0, 1]$. This generalized hypothesis returns a probability of classifying $\mathbf{x}$ as 1. By randomizing, the expected performance of the hypothesis may decrease on regular data drawn from a non-adversarial distribution, but it also may decrease the value of the queries for the attacker. Randomization in this fashion does not reduce the information available in principle to the attacker, but merely requires more work from the attacker for the information.

However, if an attacker does reverse engineer the decision boundary and (for example) discovers a region of intrusions where an IDS does not alarm, then if the decision is randomized the IDS will occasionally alarm on attack points and the attacker cannot avoid detection indefinitely. In these circumstances, the learner trades some average-case classification performance for assurance that attacks have limited duration before they are discovered.

Limiting/misleading feedback. Another potential defense is to limit the feedback given to an attacker. For example, common techniques in the spam domain include eliminating bounce emails, remote image loading, and other potential feedback channels. It is impossible to remove all feedback channels; however, limiting

feedback increases work for the attacker. In some settings, it may be possible to mislead the attacker by sending fraudulent feedback.

Actively misleading the attacker with feedback suggests an interesting battle of information between attacker and defender. In some scenarios the defender may be able to give the attacker no information via feedback, and in others the defender may even be able to give feedback that causes the attacker to come to incorrect conclusions.

2.4.2 Defending against *Causative* attacks

In *Causative* attacks, the attacker has a degree of control over not only the evaluation distribution but also the training distribution. Therefore the learning procedures we consider must be resilient against contaminated training data, as well as to the considerations discussed in the previous section. We consider two main approaches.

Robustness

The field of Robust Statistics explores procedures that limit the impact of a small fraction of deviant (adversarial) training data. In the setting of Robust Statistics, it is assumed that the bulk of the data is generated from a known model, but a small fraction of the data is selected adversarially. A number of tools exist for assessing robustness: *qualitative robustness*, the *breakdown point* of a procedure (how much data the attacker needs for arbitrary control), and the *influence function* of a procedure

(to measure the impact of contamination on the procedure). These tools can be used to design procedures that are robust against adversarial contamination of the training data. For a full treatment, see the books by Huber [1981], Hampel, Ronchetti, Rousseeuw, and Stahel [1986], and Maronna, Martin, and Yohai [2006].

Recent research has highlighted the importance of robust procedures in security and learning tasks. Wagner [2004] observes that common sensor net aggregation procedures, such as computing a mean, are not robust to adversarial point contamination, and he identifies the median and trimmed average as robust replacements. Christmann and Steinwart [2004] study robustness for learning methods that can be expressed as regularized convex risk minimization on a Hilbert space. Their results suggest that the use of loss functions with certain common properties (such as logistic loss), along with regularization, can lead to robust procedures in the sense of bounded influence. Such procedures seem to have desirable properties for secure learning.

Online prediction with experts

When the attacker has full control of the training data, the situation is more dire. If f minimizes risk on the training set, the attacker could choose A_T and A_E to make the evaluation risk approach its maximum. However, a slight change to the defender's objective yields an interesting variant of the iterated *Causative*.

Consider the case where the attacker has complete control over training data but the defender receives the advice of M *experts* who provide predictions. For example,

the defender may have M different classifiers, each of which is designed to be robust in a different way. Each classifier is an expert in this model. We construct a composite classifier that predicts based on the *advice* of the experts. We make no assumptions about how the experts perform, but we evaluate our learner's performance relative to the best expert in hindsight. The intuition behind this strategy is that the attacker must design attacks that are successful not only against a single expert, but uniformly against the set of experts. The composite learner can perform almost as well as the best one without knowing ahead of time which expert is best.

The learner forms a prediction from the M expert predictions and adapts the hypothesis based on their performance during K repetitions. At each step k of the game, the defender receives a prediction $g_m^{(k)}$ from each expert; this may be based on the data but we make no assumptions about its behavior. More formally, the k^{th} round of the expert-based prediction game is:

1. **Defender** Update function $h^{(k)} : \mathcal{Y}^M \to \mathcal{Y}$
2. **Attacker** Choose distribution $\mathbb{P}^{(k)}$
3. Evaluation:
    - Sample an instance $(\mathbf{x}^{(k)}, y^{(k)}) \sim \mathbb{P}^{(k)}$
    - Compute expert advice $\{g_m^{(k)}\}_{m=1}^M$
    - Predict $\hat{y}^{(k)} = h^{(k)}\left(g_1^{(k)}, \ldots, g_M^{(k)}\right)$.
    - Assess cost $C\left(\mathbf{x}^{(k)}, \hat{y}^{(k)}, y^{(k)}\right)$

This game has a slightly different structure from the games we present in Section 2.2.4—here the defender chooses one strategy at the beginning of the game and then in each iteration updates the function $h^{(k)}$ according to that strategy. The attacker, however, may select a new strategy at each iteration.

The setting of online expert-based prediction allows us to split risk minimization into two subproblems: (1) minimizing the loss of each expert and (2) minimizing the average *regret*—the difference between the loss of our composite learner and the loss of the best overall expert in hindsight. The other defenses we have discussed approach the first problem. Online game theory addresses the second problem: the defender chooses a strategy for updating $h^{(k)}$ to minimize regret based only on the experts's past performance. For certain variants of the game, composite predictors exist with regret $o(K)$—the average regret approaches 0 as K increases. A full description of this setting and several results appear in Cesa-Bianchi and Lugosi [2006].

2.5 Summary

In this chapter we have presented a framework for articulating a comprehensive view of different classes of attacks on machine learning systems in terms of three independent dimensions. We have analyzed the security goals and threat model for machine learning, explored the attacker's capabilities, and developed the notion of a learning game between attacker and defender. Guided by our framework and the learning game, we identify where relevant prior research fits into the framework.

Specifically, we explore the effects of different types of attacks on the systems and their defenses against these attacks.

We believe that our framework opens a number of new research directions. In particular, from the framework, we can generalize to the idea that many of the classes of attacks are dependent upon knowledge that the attacker has gained about the internal states of the learner. Thus, one potentially interesting avenue for future exploration is the idea of securing learning systems by measuring and bounding the amount of information leaked from a learning system to an attacker.

In the following chapters we examine in depth how this framework can guide attacks against and defenses of real systems.

Chapter 3

Developing Causative Attacks: Attacking SpamBayes

3.1 Introduction

This chapter demonstrates how attackers can exploit machine learning to subvert the SpamBayes statistical spam filter.[1] Our *Causative Availability* attack strategies exhibit two key differences from previous work: traditional attacks modify spam emails to evade a spam filter, whereas our attacks interfere with the training process of the learning algorithm and *modify the filter itself*; and rather than focusing only on placing spam emails in the victim's inbox, we subvert the spam filter to *remove legitimate emails* from the inbox.

[1] Material in this chapter has been previously published in the paper "Exploiting machine learning to subvert your spam filter" [Nelson et al., 2008].

We consider attackers with one of two goals: expose the victim to an advertisement or prevent the victim from seeing a legitimate message. Potential revenue gain for a spammer drives the first goal, while the second goal is motivated, for example, by an organization competing for a contract that wants to prevent competing bids from reaching their intended recipient.

An attacker may have detailed knowledge of a specific email the victim is likely to receive in the future, or the attacker may know particular words or general information about the victim's word distribution. In many cases, the attacker may know nothing beyond which language the emails are likely to use.

When an attacker wants the victim to see spam emails, a broad *dictionary attack* can render the spam filter unusable, causing the victim to disable the filter (Section 3.3.1). With more information about the email distribution, the attacker can select a smaller dictionary of high-value features that are still effective. When an attacker wants to prevent a victim from seeing particular emails and has some information about those emails, the attacker can target them with a *focused attack* (Section 3.3.2).

Our attacks target the learning algorithm underlying several spam filters, including SpamBayes (spambayes.sourceforge.net), BogoFilter (bogofilter.sourceforge.net), and the machine learning component of SpamAssassin (spamassassin.apache.org)—the primary difference between the learning elements of these three filters is in their tokenization methods. Of these, we choose SpamBayes because it uses a pure ma-

chine learning method, it is familiar to the academic community [Meyer and Whateley, 2004], and it is popular with over 700,000 downloads. Although we specifically attack SpamBayes, the widespread use of its statistical learning algorithm suggests that other filters may also be vulnerable to similar attacks. However, some filters, such as SpamAssassin, use the learner only as one component of a broader filtering strategy.

Our experimental results confirm that this class of attacks presents a serious concern for statistical spam filters. A dictionary attack can make a spam filter unusable when controlling just 1% of the messages in the training set, and a well-informed focused attack can remove the target email from the victim's inbox 90% of the time.

3.2 Background

3.2.1 Training model

SpamBayes produces a classifier from labeled examples to label future emails. The labels are *spam* (bad, unsolicited email), *ham* (good, legitimate email), and *unsure* (SpamBayes isn't confident one way or the other). The classifier learns from a labeled *training set* of ham and spam emails.

Email clients use these labels in different ways—some clients filter email labeled as *spam* and *unsure* into "Spam-High" and "Spam-Low" folders, respectively, while other clients only filter email labeled as *spam* into a separate folder. Since the typical

user reads most or all email in their inbox and rarely (if ever) looks at spam folders, the *unsure* labels can be problematic. If *unsure* messages are filtered into a separate folder, users may periodically read the messages in that folder to avoid missing important email. If instead *unsure* messages are not filtered, then the user faces those messages when checking the email in their inbox. Too much *unsure* email is almost as troublesome as too many false positives (ham labeled as *spam*) or false negatives (spam labeled as *ham*). In the extreme, if everything is labeled *unsure* then the user obtains no time savings at all from the filter.

In our scenarios, an organization uses SpamBayes to filter incoming email for multiple users[2] and trains on everyone's received email. SpamBayes may also be used as a personal email filter, in which case the presented attacks are likely to be equally effective.

To keep up with changing trends in the statistical characteristics of both legitimate and spam email, we assume that the organization retrains SpamBayes periodically (e.g., weekly). Our attacks are not limited to any particular retraining process; they only require that the attacker can introduce attack data into the training set somehow (the contamination assumption).

[2]We use the terms *user* and *victim* interchangeably for either organization or individual; the meaning will be clear from context.

3.2.2 The contamination assumption

We assume that the attacker can send emails that the victim will use for training—the *contamination assumption*—but incorporate two significant restrictions: attackers may specify arbitrary email bodies but not headers, and attack emails are always trained as spam and not ham. We examine the implications of the contamination assumption in the remainder of this chapter.

A real attacker has many opportunities to contaminate the training set. Consider the following alternatives. If the victim periodically retrains on all email, any email the attacker sends will be used for training. If the victim trains on only manually labeled email, the attack emails will still be included as spam because they genuinely are spam. Even if the victim retrains only on mistakes made by the filter, it may not be difficult to design emails that perform our attacks and are misclassified by the victim's current filter.

We do not consider the possibility that a user might inspect training data to remove attack emails. It seems unrealistic that any but the most sophisticated user would have any idea whether a particular spam email might damage the performance of the filter. If the user has heuristics for detecting the attack emails we describe later, our attacks could easily be adjusted to evade simple heuristics such as email size or word distributions. We avoid pursuing this arms race here.

Our focus on attack emails trained as spam should be viewed as a restriction and not a necessary condition for the success of the attacks—the ability to use attack

emails trained as ham could enable additional attacks that place spam in a user's inbox.

3.2.3 SpamBayes learning method

SpamBayes classifies using token scores based on a simple model of spam status proposed by Robinson [Meyer and Whateley, 2004; Robinson, 2003], based on ideas by Graham [Graham, 2002] together with Fisher's method for combining independent significance tests [Fisher, 1948]. Intuitively, SpamBayes learns how strongly each word indicates *ham* or *spam* by counting in how many of each type of email that word appears. When classifying a new email, SpamBayes looks at all of its words and uses a statistical test to decide whether they indicate one label or the other with sufficient confidence; if not, SpamBayes returns *unsure*.

SpamBayes tokenizes the header and body of each email before constructing token spam scores. Robinson's method assumes that each token's presence or absence in an email affects that email's spam status independently from other tokens. For each token w, SpamBayes computes the *raw token score*

$$P_R(w) \quad = \quad \frac{N_H N_S(w)}{N_H N_S(w) + N_S N_H(w)} \tag{3.1}$$

from the counts N_S, N_H, $N_S(w)$, and $N_H(w)$—the number of spam emails, ham emails, spam emails that include w and ham emails that include w.

Robinson smooths $P_R(w)$ through a convex combination with a prior belief b, weighting the quantities by $N(w)$ (the number of training emails with w) and s

(chosen for strength of prior), respectively. This leads to the *token score*

$$P_S(w) \;=\; \frac{s}{s+N(w)}\,b + \frac{N(w)}{s+N(w)}\,P_R(w)\;. \tag{3.2}$$

For a new message $\mathbf{x}$, Robinson uses Fisher's method to combine the spam scores of the most significant tokens into a *message score*, or *spam score*. SpamBayes uses at most 150 tokens from $\mathbf{x}$ with scores furthest from 0.5 and outside the interval $[0.4, 0.6]$. We call this set $\nu(\mathbf{x})$. The message score is

$$I(\mathbf{x}) \;=\; \frac{1+H(\mathbf{x})-S(\mathbf{x})}{2} \;\in\; [0,1]\;, \tag{3.3}$$

$$H(\mathbf{x}) \;=\; 1-\chi^2_{2n}\left(-2\sum_{w\in\nu(\mathbf{x})}\log P_S(w)\right)\;, \tag{3.4}$$

where $\chi^2_{2n}(\cdot)$ denotes the cumulative distribution function of the chi-square distribution with $2n$ degrees of freedom. $S(\mathbf{x})$ is defined like $H(\mathbf{x})$ but with $P_S(w)$ replaced by $1-P_S(w)$. SpamBayes predicts by thresholding against two user-tunable thresholds θ_0 and θ_1, with defaults $\theta_0 = 0.15$ and $\theta_1 = 0.9$: SpamBayes predicts *ham*, *unsure*, or *spam* if I falls into the interval $[0,\theta_0]$, $(\theta_0,\theta_1]$, or $(\theta_1,1]$, respectively.

The inclusion of an *unsure* category in addition to *spam* and *ham* prevents us from purely using *ham-as-spam* and *spam-as-ham* misclassification rates (false positives and false negatives, respectively) for evaluation. We must also consider *spam-as-unsure* and *ham-as-unsure* emails. Because of the practical effects on the user's time and effort mentioned in Section 3.2.1, *ham-as-unsure* misclassifications are nearly as bad for the user as *ham-as-spam*.

3.3 Attacks

Our focus is on *Causative Availability* attacks, which manipulate the filter's training data to increase false positives. We consider both *Indiscriminate* and *Targeted* attacks. In *Indiscriminate* attacks, enough false positives force the victim to disable the spam filter, or at least frequently search through *spam*/*unsure* folders to find legitimate messages that were filtered away. In either case, the victim is forced to view more spam. In *Targeted* attacks, the attacker does not disable the filter but surreptitiously prevents the victim from receiving certain types of email. For example, a company may wish to prevent its competitors from receiving email about a bidding process in which they are all competing.

3.3.1 Dictionary attacks

Our first attack is an *Indiscriminate* attack—the attacker wants to cause a large number of false positives so that the user loses confidence in the filter and must manually sort through spam and ham emails. The idea is to send *attack emails* that contain many words likely to occur in legitimate email. When the victim trains SpamBayes with these attack emails marked as *spam*, the words in the attack emails will have higher spam score. Future legitimate email is more likely to be marked as *spam* if it contains words from the attack email.

When the attacker lacks specific knowledge about the victim's email, one simple attack is to include an entire dictionary of the English language. This technique is the

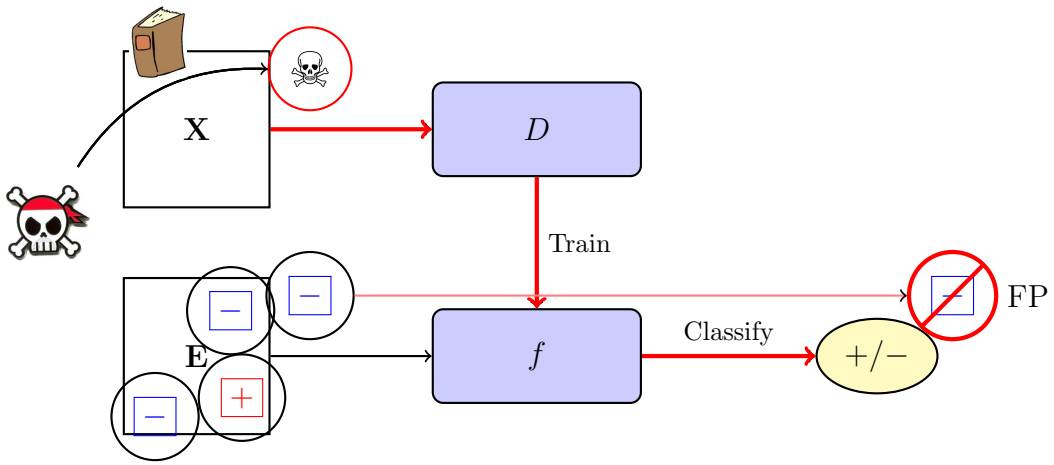


Figure 3.1: Dictionary attack against SpamBayes. The attacker sends dictionary attack messages that the victim trains on, causing the learned classifier f to incorrectly reject many benign emails, resulting in *Indiscriminate* false positives.

basic *dictionary attack*. See Figure 3.1 for a visual representation. A further refinement uses a word source with a distribution closer to the victim's email distribution. For example, a large pool of *Usenet* newsgroup postings may have colloquialisms, misspellings, and other "words" not found in a dictionary; furthermore, using the most frequent words in such a corpus may allow the attacker to send smaller emails without losing much effectiveness. For more details on the datasets we use, see Section 3.4.1.

3.3.2 Focused attack

Our second attack is a *Targeted* attack—the attacker has some knowledge of a specific legitimate email to target for filtering. If the attacker has exact knowledge of the target email, placing all of its tokens in attack emails produces an optimal attack. Realistically, the attacker has partial knowledge about the target email and can guess only some of its tokens to include in attack emails. We model this knowledge by

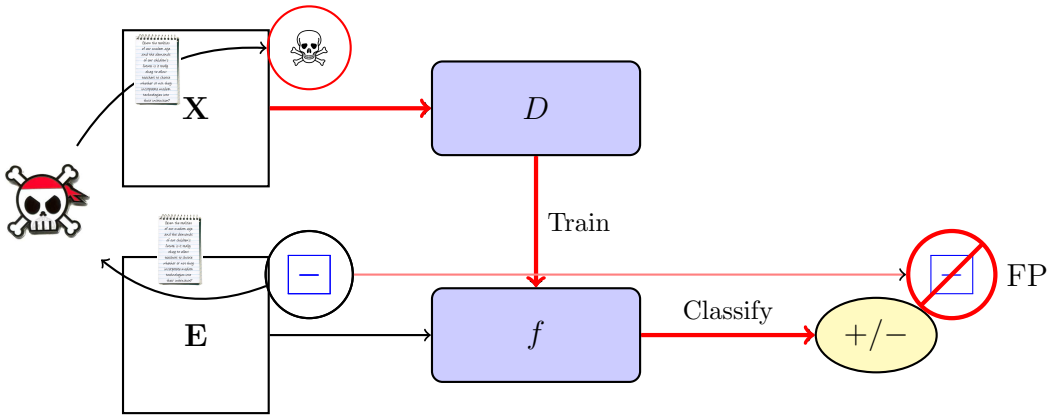


Figure 3.2: Focused attack against SpamBayes. The attacker sends attack messages containing words from a specific target message. When SpamBayes trains on them, the resulting classifier f learns to reject the target message (a *Targeted* false positive).

letting the attacker probabilistically guess tokens from the target email to include. When SpamBayes trains on this attack email, the spam scores of the targeted tokens increase, so the target message is more likely to be filtered as *spam*. This is the *focused attack* (see Figure 3.2). For example, consider a malicious contractor wishing to prevent the victim from receiving messages with competing bids. The attacker sends spam emails to the victim with words such as the names of competing companies, their products, and their employees. The bid messages may even follow a common template, making the attack easier to craft.

The attacker may have different levels of knowledge about the target email. In the extreme case, the attacker might know the exact content of the target email and use all of its words. More realistically, the attacker only guesses a fraction of the email's content. In either case, the attack email may include additional words as well.

The focused attack is more concise than the dictionary attack because the attacker

has detailed knowledge of the target and no reason to affect other messages.

3.3.3 Optimal attack function

The dictionary and focused attacks can be seen as two instances of a common attack in which the attacker has different amounts of knowledge about the victim's email. Without loss of generality, suppose the attacker generates only a single attack message $\mathbf{a}$. The victim adds it to the training set, trains, and classifies a new message $\mathbf{x}$. Both $\mathbf{a}$ and $\mathbf{x}$ are indicator vectors, where the i^{th} component is true if word i appears in the email. The attacker also has some (perhaps limited) knowledge of the next email the victim will receive. This knowledge can be represented as a distribution $\mathbf{p}$—the vector of probabilities that each word appears in the next message.

The goal of the attacker is to choose an attack email $\mathbf{a}$ that maximizes the *expected spam score:*

$$\max_{\mathbf{a}} \; \mathbb{E}_{\mathbf{x}\sim\mathbf{p}} \left[I_{\mathbf{a}}(\mathbf{x}) \right].$$

In other words, the goal of the attack is to maximize the expectation of $I_{\mathbf{a}}$ (Equation (3.3) with the attack message $\mathbf{a}$ in the spam training set) of the next legitimate email $\mathbf{x}$ drawn from distribution $\mathbf{p}$. In order to describe the optimal attack under this criterion, we make two observations. First, the token scores of distinct words do not interact; that is, adding a word w to the attack does not change the score $P_S(u)$ of some different word $u \neq w$. Second, it can be shown that I is monotonically non-decreasing in each $P_S(w)$. Therefore the best way to increase $I_{\mathbf{a}}$ is to include

additional words in the attack message.

Now let us consider specific choices for the next email's distribution $\mathbf{p}$. First, if the attacker has little knowledge about the words in target emails, we model this by setting $\mathbf{p}$ to be uniform over all vectors $\mathbf{x}$ representing emails. We can optimize the expected message score by including *all possible words* in the attack email. This *optimal attack* is infeasible in practice but can be simulated: one approximation includes all words in the victim's primary language, such as an English dictionary. Using a dictionary in this way yields the dictionary attack.

Second, if the attacker has specific knowledge of a target email, we can represent this by setting p_i to 1 if and only if the i^{th} word is in the target email. The optimal attack still maximizes the expected message score, but a more compact attack that is also optimal is to include all of the words in the target email. This approach is the focused attack.

The attacker's knowledge usually falls between these extremes. For example, the attacker may use information about the distribution of words in English text to make the attack more efficient, such as characteristic vocabulary or jargon typical of emails the victim receives. Either of these results in a distribution $\mathbf{p}$ over words in the victim's email. Using this distribution, it should be possible to derive an optimal constrained attack, which would be an interesting line of future work.

3.4 Experiments

3.4.1 Experimental method

In our experiments we use the Text Retrieval Conference (TREC) 2005 spam corpus [Cormack and Lynam, 2005], which contains 92,189 emails (52,790 spam and 39,399 ham). The TREC corpus is based on the Enron email corpus [Klimt and Yang, 2004], which consists of emails subpoenaed as evidence in the Enron trial and made public, as well as additional spam email data. This corpus has several strengths: it comes from a real-world source, it has a large number of emails, and its creators took care that the added spam does not have obvious artifacts to differentiate it.

We use two sources of dictionary words. First, we use the GNU `aspell` English dictionary version 6.0-0, containing 98,568 words. Second, we use the 90,000 most common words from a subset of Usenet English postings [Shaoul and Westbury, 2007]. The overlap between these two dictionaries is approximately 61,000 words.

We restrict the attacker to have limited control over the headers of attack emails. We implement this assumption by using the entire header from a different randomly selected spam email from TREC for each attack email, taking care to ensure that the content-type and other Multipurpose Internet Mail Extensions (MIME) headers correctly reflect the attack message body.

We measure the effect of each attack by randomly choosing a 10,000-message inbox that is 50% ham and 50% spam and comparing classification performance of the

Parameter	Dictionary Attack	Focused Attack
Training set size	2,000, 10,000	2,000, 10,000
Spam prevalence	0.50, 0.75, 0.90	0.50, 0.75, 0.90
Attack fraction	0.001, 0.005, 0.01, 0.02, 0.05, 0.10	0.001, 0.005, 0.01, 0.02, 0.05, 0.10
Validation folds	10	10
Test set size	1,000	N/A
Target Emails	N/A	20

Table 3.1: Parameters for our SpamBayes attack experiments.

control and compromised filters using ten-fold cross-validation. In cross-validation, we partition the data into ten subsets and perform ten train-test epochs. During the i^{th} epoch, the i^{th} subset is set aside as a test set and the remaining nine subsets are used for training. Each email from our original 10,000-message dataset serves independently as both training and test data.

In the following sections, we show the effect of our attacks on test sets of held-out messages. Because our attacks are designed to cause ham to be misclassified, we only show their effect on ham messages; their effect on spam is marginal. Our graphs do not include error bars since we observed that the variation on our tests was small. See Table 3.1 for our experimental parameters. We found the size of the training set and spam prevalence in the training set to have minimal impact on the results, so we present only the results of 10,000-message training sets at 50% spam prevalence.

3.4.2 Dictionary attack results

We examine dictionary attacks as a function of the percent of attack messages in the training set. Figure 3.3 shows the misclassification rates of three dictionary attack variants averaging over ten-fold cross-validation. The optimal attack quickly causes the filter to label all ham emails as *spam*. The Usenet dictionary attack causes significantly more ham emails to be misclassified than the `aspell` dictionary attack, since it contains common misspellings and slang terms that are not present in the `aspell` dictionary (the overlap between the `aspell` and Usenet dictionaries is around 61,000 words). These variations of the attack require relatively few attack emails to significantly degrade SpamBayes's accuracy. By 101 attack emails (1% of 10,000), the accuracy falls significantly for each attack variation; at this point most users will gain no advantage from continued use of the filter.

To be fair, although the attack emails make up a small percentage of the *number of messages* in a contaminated inbox, they make up a large percentage of the *number of tokens*. For example, at 204 attack emails (2% of the messages), the Usenet attack includes approximately 6.4 times as many tokens as the original dataset and the `aspell` attack includes 7 times. An attack with fewer tokens likely would be harder to detect; however, the number of messages is a more visible feature. It is of significant interest that so few attack messages can degrade a widely-deployed filtering algorithm to such a degree.

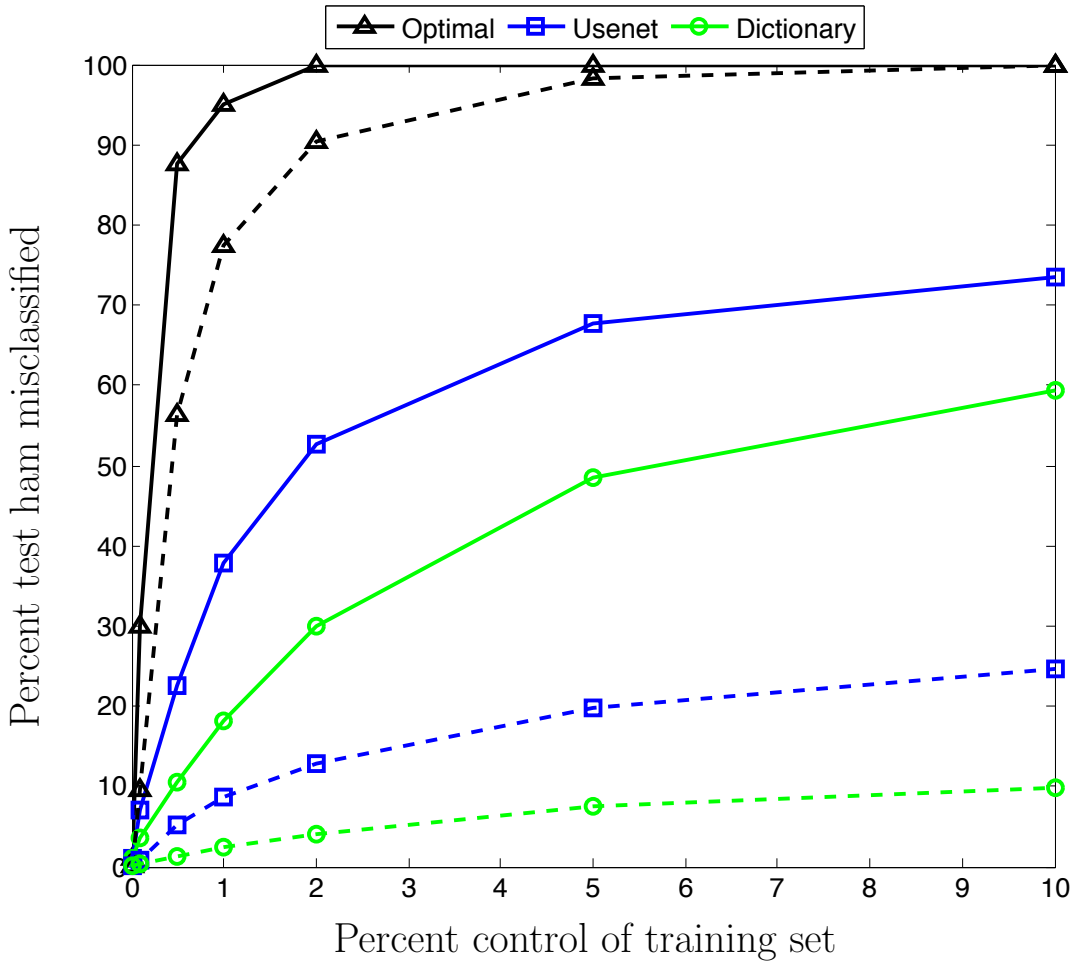


Figure 3.3: Three dictionary attacks on initial training set of 10,000 messages (50% spam). We plot percent of ham classified as *spam* (dashed lines) and as *spam* or *unsure* (solid lines) against the attack as percent of the training set. We show the optimal attack (black △), the Usenet dictionary attack (blue □), and the `aspell` dictionary attack (green ○). All render the filter unusable with as little as 1% control (101 messages).

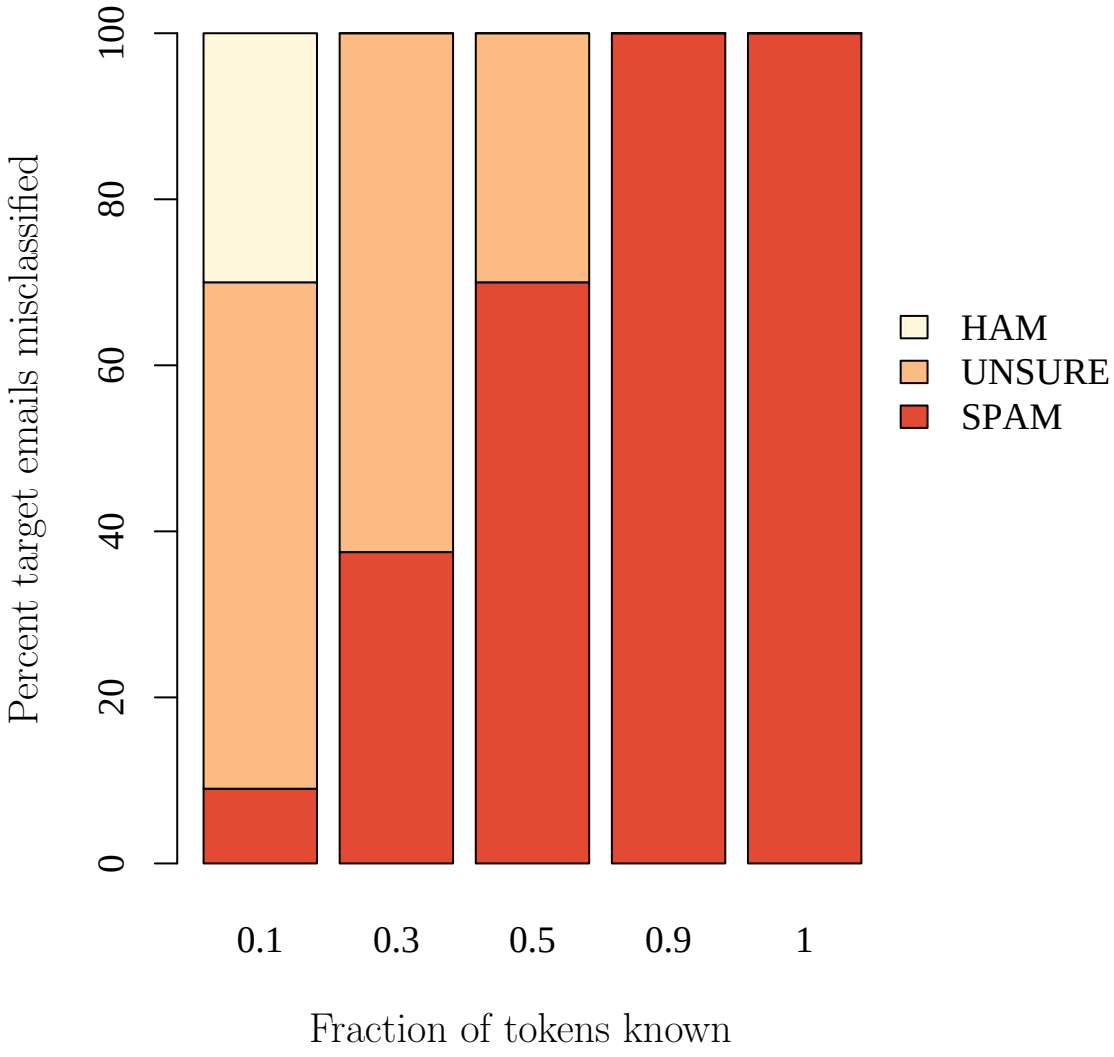


Figure 3.4: Effect of the focused attack as a function of the fraction of target tokens known. Each bar depicts the fraction of target emails classified as *spam*, *ham*, and *unsure* after the attack. The initial inbox contains 10,000 emails (50% spam).

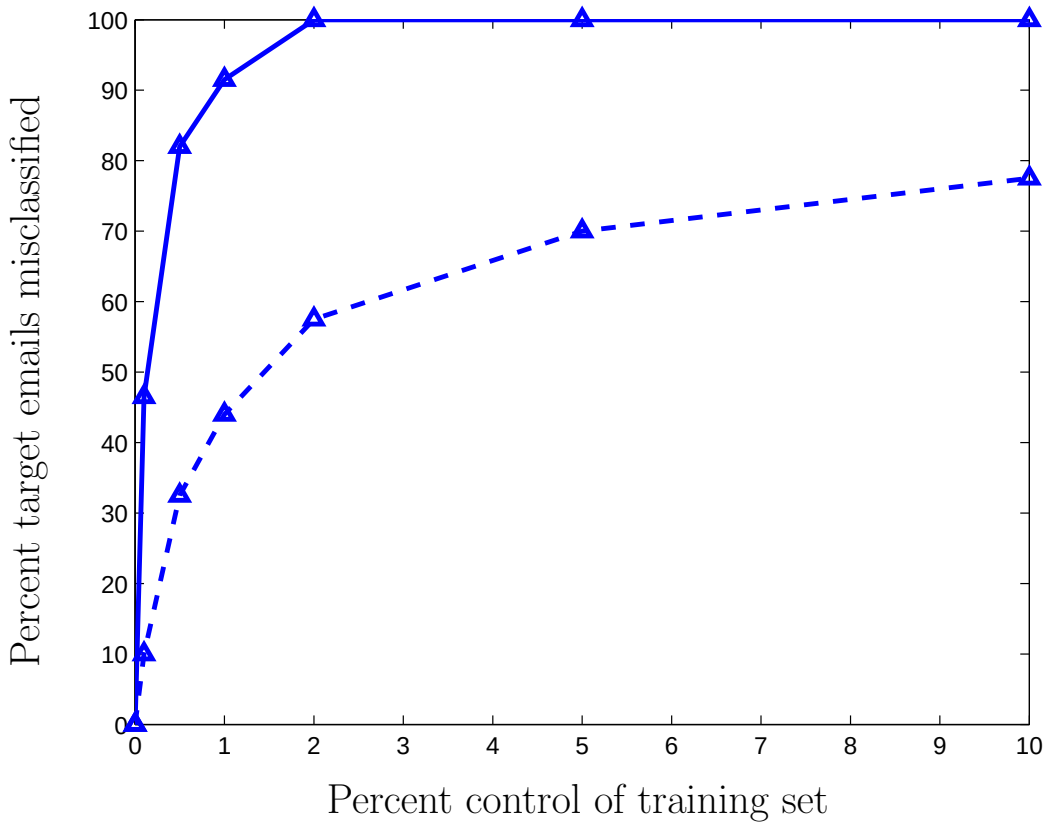


Figure 3.5: Effect of the focused attack as a function of the number of attack emails with a fixed fraction (p_w=0.5) of tokens that the attacker knows. The dashed line shows the percentage of target ham messages misclassified as *spam* after the attack, and the solid line the percentage of targets that are misclassified as *unsure* or *spam* after the attack. The initial inbox contains 10,000 emails (50% spam).

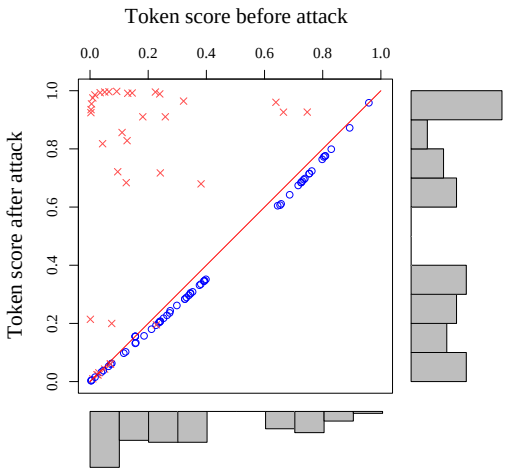
3.4.3 Focused attack results

For the focused attack, we randomly select 20 ham emails from the TREC corpus to serve as the target emails before creating the 10,000-message inbox. During each fold of cross-validation, we perform 20 focused attacks, one for each email, so our results average over 200 different runs.

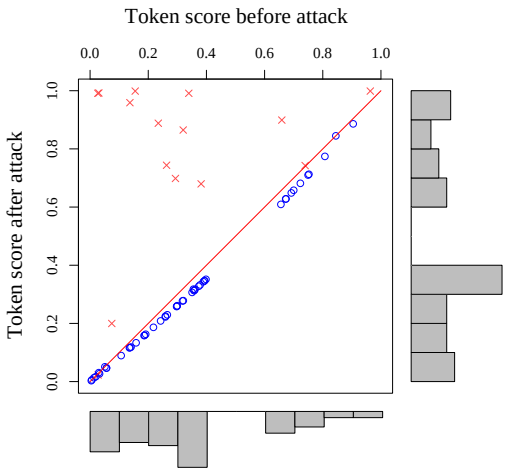
In Figure 3.4, we examine the effectiveness of the attack when the attacker has increasing knowledge of the target email by simulating the process of the attacker guessing tokens from the target email. For this figure, there are 300 attack emails—16% of the original number of training emails. We assume that the attacker knows a certain fraction p_w of tokens in the target email, with $p_w \in \{0.1, 0.3, 0.5, 0.9\}$—the x-axis of Figure 3.4. The y-axis shows the proportion of the 20 targets classified as *ham*, *unsure* and *spam*. As expected, the attack is increasingly effective as p_w increases. If the attacker knows 50% of the tokens in the target, classification changes on all of the target emails with a 75% rate of classifying as *spam*.

In Figure 3.5, we examine the attack's effect on misclassifications of the target emails as the number of attack messages increases. In this figure, we fix the fraction of tokens known at 0.5. The x-axis is the number of messages in the attack given as percent of the training set, and the y-axis is the percent of messages misclassified. With 100 attack emails and an initial mailbox size of 5,000 (2%), the target email is misclassified nearly 100% of the time.

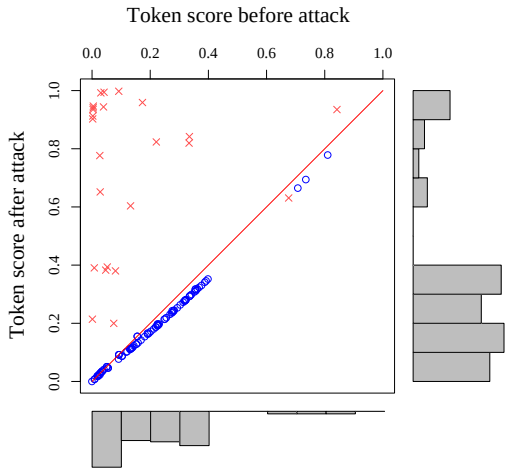
We gain more insight by examining the attack's effect on three representative



(a) Misclassified as *spam*



(b) Misclassified as *unsure*



(c) Correctly classified *ham*

Figure 3.6: Effect of the focused attack on three representative target emails; red ×'s are attacked tokens and blue ○'s are other tokens. See the text for a full explanation.

emails (see Figure 3.6). Each of the panels in the figure represents a single target email from each of three attack results: ham misclassified as *spam* (Figure 3.6a), ham misclassified as *unsure* (Figure 3.6b), and ham correctly classified as *ham* (Figure 3.6c). Each point is a token in the email. The x-axis is the token spam score (from Equation (3.2)) before the attack, and the y-axis is the token score after the attack (0 means *ham* and 1 means *spam*). The red $\times$'s are tokens included in the attack (known by the attacker) and the blue $\bigcirc$'s are tokens not in the attack. The histograms show the distribution of token scores before the attack (at bottom) and after the attack (at right).

Any point above the line $y = x$ increased due to the attack and any point below is a decrease. For these experiments the attacker knows 10% of the email's tokens. In these graphs we see that tokens included in the attack typically increase significantly while those not included decrease slightly. Since the increase in score is more significant for included tokens than the decrease in score for excluded tokens, the attack has substantial impact even when the attacker has a low probability of guessing tokens, as seen in Figure 3.4. Furthermore, the before/after histograms in Figure 3.6 provide a direct indication of the attack's success.

3.5 Summary

In this chapter, we show that an adversary can effectively disable the SpamBayes spam filter with relatively little system state information and relatively limited control

over training data. Our Usenet dictionary attack causes misclassification of 36% of ham messages with only 1% control over the training messages, rendering SpamBayes unusable. Our focused attack changes the classification of the target message 60% of the time with only a 30% chance of guessing each of the target's tokens.

The attacks we present in this chapter are similar to the correlated outlier attacks suggested by Newsome, Karp, and Song [2006]. They speculate briefly about applying such an attack to spam filters; however, several of their assumptions about the learner are not appropriate for learners such as SpamBayes. For example, they assume that the learner uses only features indicative of the positive class. Furthermore, although they present insightful analysis, they do not evaluate the correlated outlier attack against a real system. Our attacks use similar ideas, but we develop and test them on a real system. We also explore the value of information to an attacker with our Usenet dictionary attack and focused attack.

Our attacks should also work against other spam filtering systems based on similar learning algorithms, such as BogoFilter and the Bayesian component of SpamAssassin although their effect may vary (and SpamAssassin uses more components than just the learning algorithm, so the overall effect of our attacks may be small). Similar techniques may also be effective against other learning systems, such as those used for worm or intrusion detection.

Chapter 4

Exploratory Attack Defense: LDA for Virus Detection

4.1 Introduction

In this chapter, we examine defensive strategies against an *Exploratory Integrity* attack that combine high-level global information with low-level local information.[1] The attack we consider is an email virus attack with traces from nine real viruses that use various strategies to evade detection. The attacker wants to infect end-user machines; we study an organizational setting where the administrator needs to detect which users have infected machines so they can be quarantined and cleaned. Because we are attempting to detect user-level compromise, we expect user-specific features to

[1]Material in this chapter has been previously published in the paper "User model transfer for email virus detection" [Barreno et al., 2006a].

be useful in distinguishing normal users from compromised ones. As an organization with many users, however, the defender also has substantial information about global behaviors and patterns. Here we explore a technique for leveraging both these types of information to detect virus-infected users.

The motivation for using machine learning for virus detection is clear. Most successful viruses and worms spread via email [Sophos Corporation, 2005]. Traditional network- and host-based virus scanners rely on manually crafted signatures and heuristics and have difficulty detecting novel viruses.[2] This creates a window of vulnerability each time a virus is released. For example, anti-virus vendors took over seven hours on average to generate a virus signature for the MyDoom.BB outbreak [Seltzer, 2005]. We list the response times in Table 4.1 to highlight their wide range.

In contrast, machine learning techniques automatically model behavioral features of normal (or abnormal) email traffic, allowing them to detect unknown attacks by recognizing subtle deviations from normal activity. Most existing machine learning approaches to virus detection use either global or per-user models.

Global models generalize across all users (or network events, etc.) to take advantage of the full scope of data available. They often benefit from plentiful training data but their accuracy may be limited by variations between users. Network intrusion detection systems usually build global models, in particular for anomaly detection [Es-

[2]We do not distinguish between email-based worms and viruses.

Vendor	Time (hh:mm)
ClamAV	0:00
Sophos	0:58
Trend Micro	2:03
Fortinet	2:16
F-Prot	2:41
McAfee	2:50
eTrust-CA	3:28
Symantec	4:04
Command	4:42
Virusbuster	4:56
Trend Micro	5:10
Quickheal	6:09
eTrust-VET	7:29
AntiVir	8:07
Ikarus	8:27
Dr. Web	9:01
Proland	9:17
Panda	9:39
RAV	9:41
BitDefender	9:54
Norman	10:18
Dr. Web	10:52
AVG	12:03
Kaspersky	14:16
F-Secure	16:44
Avast	17:21

Table 4.1: Response times of anti-virus vendors in hours to the MyDoom.BB virus, February 16–17, 2005. Times are given starting from the first response. (The exact time of the outbreak is unknown.) Some vendors are repeated because their first response incorrectly identified the virus and the second corrects the identification. This table is derived from information given by Seltzer [2005].

kin, Arnold, Prerau, Portnoy, and Stolfo, 2002; Lazarevic, Ertöz, Kumar, Ozgur, and Srivastava, 2003]. These systems build a model of typical user or network behavior and flag activity that falls outside the learned model.

Local models treat each user's behavior independently, as is common in personal spam detection systems [Meyer and Whateley, 2004; Robinson, 2003]. Separate per-user models can be more accurate in the long run but suffer from a lack of training data when a new user enters the system.

Some existing approaches also combine global and per-user information in their models. The Email Mining Toolkit uses several machine learning methods, including naive Bayes classification and social network analysis, on both global and per-user levels to detect email-borne viruses. Using a back-and-forth search heuristic, it finds agreements between the models to classify sequences of malicious emails [Stolfo, Hershkop, Hu, Li, Nimeskern, and Wang, 2006].

Another combined system, APE, uses both a global model and per-user models in real-time to provide dynamic containment of worms and viruses. It uses a global model to flag suspicious messages, which are then classified by per-user models [Martin, Sewani, Nelson, Chen, and Joseph, 2005; Martin, 2005].

Our approach uses *Latent Dirichlet Allocation (LDA)*, a probabilistic model that combines global and per-user information, gracefully transitioning between them as more user data becomes available [Blei, Ng, and Jordan, 2003].

Single	Window
CharsInSubject	NumToAddrInWindow
LinksInEmail	MeanCharInSubject
AvgWordLength	MeanWordsInBody
WordsInBody	VarCharInSubject
WordsInSubject	VarWordsInBody

Table 4.2: The "single" column features are derived from one email, while the "window" column is computed from the five most recent emails.

4.2 Using LDA for email

4.2.1 Features for email

We represent emails using the features in Table 4.2. Those in the "Single" column are computed from a single email, while those in the "Window" column are computed based on a sliding window. All features are modeled with the Gaussian distribution except for LinksInEmail, for which we use the Binomial distribution. We do not use message headers, attachment information, or language-based features, such as word frequency. Instead we focus on simple properties of the email text and user sending patterns. Our feature set is based on a previous study of feature selection for email anti-virus systems [Martin et al., 2005].

The dataset version we use for these experiments does not contain attachment information [Shetty and Adibi]. Attachment features are useful but are not silver bullets, since not all viruses require attachments to propagate. For example, the BubbleBoy virus spreads via a script embedded in an email. When the email is viewed by a vulnerable mail client, it infects the system.

Some features we initially considered, such as "Number of From addresses from one sender in a window," trivially classify large portions of the data. However, in all these cases the feature in question could easily be spoofed by a virus to avoid detection. We omit such features, so our results are somewhat pessimistic.

4.2.2 Latent Dirichlet allocation

Our system is based on the premise that different users exhibit many of the same canonical behaviors when sending email, but in different proportions. Likewise, viruses all spread from host to host in some manner, so even new viruses will have some behaviors in common with known viruses. We use the Latent Dirichlet Allocation (LDA) model [Blei et al., 2003] to combine user-specific training with behavioral information learned from the full population of users and known viruses.

LDA is a probabilistic model that represents items (in our context, emails) in terms of *topics* that group items by shared characteristics. We represent an email as a vector of features. Due to our choice of features, a topic corresponds to a type of user behavior or style of email (e.g., we have observed a topic that contains primarily long forwarded emails and another that has short bodies and empty or one-word subject lines). A topic groups emails that share characteristics described by our feature set. A user is represented as a multinomial distribution over topics. In other words, LDA extracts common behaviors and represents each user as a mixture of those behaviors.

The remainder of this section describes mixture models and LDA in the context

of email virus detection. Figure 4.1 shows graphical model representations [Jordan, 2004] of the joint probability distributions of these models.

4.2.3 Variables and notation

Let T be the number of topics (chosen as a model parameter) and F be the number of features. In Figure 4.1, $\mathbf{x}$ is a vector of F components, z and α are scalar values, ϕ is a vector of T components, and β is a $T \times F$ matrix of parameters. U is the number of users, and E is the number of emails sent per user.[3] Variables inside the rectangular plates are replicated, so each model depicts a total of $E \times U$ variables $\mathbf{x}$, and so on; we do not distinguish these notationally.

4.2.4 Mixture models

A mixture model is a statistical tool for modeling datasets containing multiple subpopulations, each with a simple distribution (such as the Gaussian distribution). Mixture models can be used for global or local modeling. In Figure 4.1a we show an example of a *global mixture model (GMM)* for email. A corpus of messages is represented by a single mixture model, in which each topic is a subpopulation. Each email $\mathbf{x}$ is assigned a topic z, which selects the parameters of the email's feature distributions. There is a global distribution ϕ over topics and a global set of parameters β for feature distributions. There is no differentiation between users in this model.

[3]For convenience our discussion assumes that each user sends the same number of emails, but this assumption is not important and is in fact not true for our experimental data.

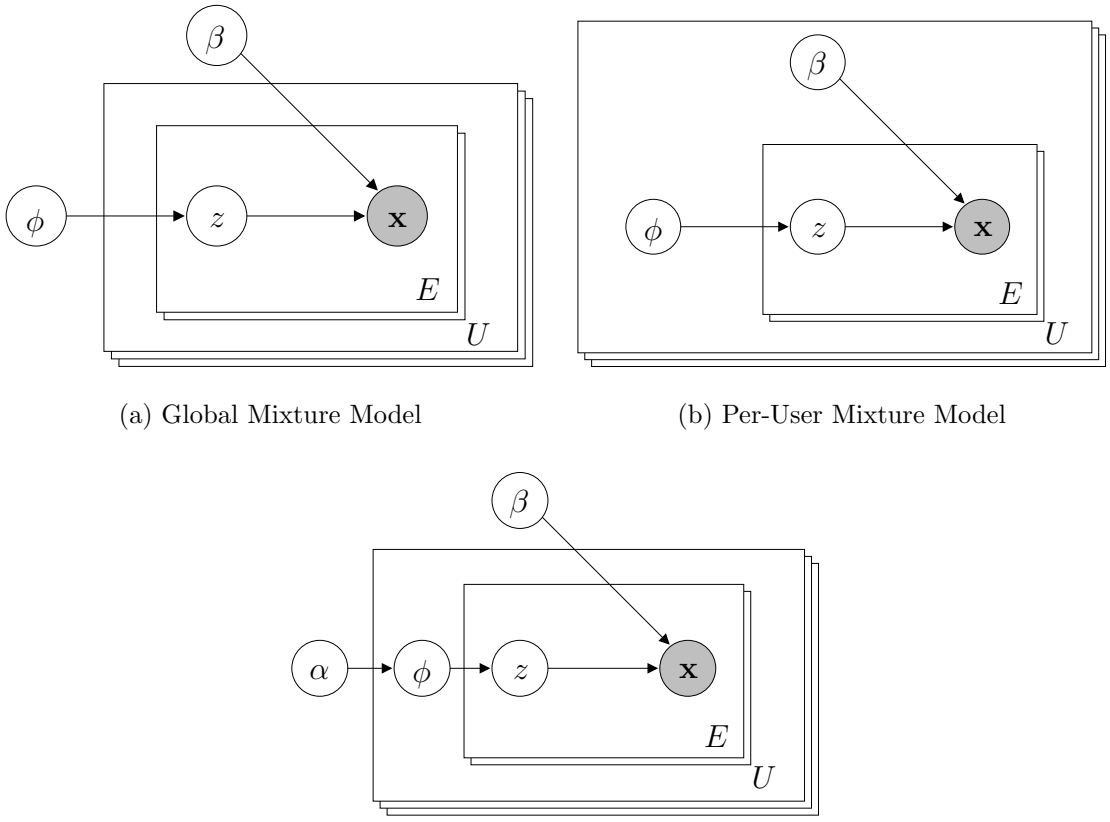
Another approach is to use a separate *per-user mixture model (PMM)* for each user, as shown in Figure 4.1b. Here again each topic is a subpopulation, but now each user has their own feature parameters β and topic distribution ϕ. There is no sharing of information across users in this model.

4.2.5 LDA: Modeling email users

The graphical model representation of LDA appears in Figure 4.1c. Like both mixture models, each email $\mathbf{x}$ belongs to a particular topic z that determines the distributions for the features of $\mathbf{x}$. Like the global model, LDA has one shared β for all users; like the local model, each user has a separate ϕ.

The LDA model can be described as a generative process, with a global prior α on topics from which a multinomial parameter ϕ is drawn for each user. When a user sends an email, the email's topic z is drawn from ϕ, and then an email $\mathbf{x}$ is produced according to the corresponding distribution from β. Each row of β contains the parameters of one topic's distribution.

Exact inference in LDA is intractable. For our implementation we use a variational approach with surrogate parameters for approximate inference and parameter estimation, as in the original LDA paper [Blei et al., 2003].



(a) Global Mixture Model

(b) Per-User Mixture Model

(c) LDA Model

Figure 4.1: Mixture models in the graphical model formalism [Jordan, 2004]. Each node is a random variable in the model and the graph represents their joint probability distribution; rectangles indicate replication. Each model represents U users each sending E emails; an email $\mathbf{x}$ has topic z, drawn from distribution ϕ, with feature parameters β. In LDA, α is a prior over distributions. In each model there is one topic z per email; the most significant difference between models is whether ϕ and β are unique for each user (inside the U rectangle) or shared globally (outside).

Extending LDA for features

In the original presentation of LDA, β holds parameters of multinomial distributions. In a population of users modeled as collections of emails, however, a multinomial is not rich enough to represent an email.[4]

We extend LDA to model each email $\mathbf{x}$ as a vector of features. The distribution of an email is a fully factored naive Bayes model: given the topic, the features (which can have different distributions) are independent. Other models for the joint distribution of an email's features could also be used in place of naive Bayes.

Shared global behaviors

Information passes between users via the global parameters α and β. The prior distribution on user parameters ϕ is Dirichlet with parameter α, and estimating α from training data yields a prior from which to draw ϕ parameters for new users. To understand how LDA balances between the prior and empirical data, assume that the email topics z are known. Then for each user we can count emails from each topic, and the posterior distribution of ϕ is Dirichlet with parameter $\alpha + \mathbf{t}$, where $\mathbf{t}$ is a T-length vector that counts emails from each topic [Bickel and Doksum, 2001]. As the number of emails increases, the expectation of ϕ smoothly transitions from the prior to the empirical distribution. Although topics are not actually observed, this provides intuition into how empirical data eventually outweighs the Dirichlet prior.

[4]Note that a document in the original LDA paper corresponds to one of our users, while one of their words corresponds to one of our email messages.

4.2.6 Classification

We use a generative approach for classification. We train two models; one learns normal behavior while the other learns virus behavior. The normal model computes topic distributions for a set of normal email users, while the virus model computes topic distributions for a set of known viruses; both models encompass the behavior of several users (whether normal users or viruses). To classify an email, we compute the likelihood that the email would be generated under each model and choose the class with the higher likelihood. The likelihood depends on which user sent the email because each user has a different topic distribution ϕ. For the normal model, we use the true sender's ϕ to compute likelihood; for the virus model, we estimate which virus is most likely to generate the email and use ϕ for that virus to compute likelihood.

4.3 Experiments

We perform experiments to compare LDA's ability to learn a new user's behavior with models that use only global or local information. The learners we compare against are a global mixture model (GMM) and per-user mixture model (PMM) as described in Section 4.2.4, as well as a linear *support vector machine (SVM)*. We use the SVM-Light software [Joachims, 1999].

4.3.1 Datasets

These experiments use a version of the Enron email corpus provided in the form of a database [Cohen; Shetty and Adibi] and emails generated by real-world viruses. We use existing email traces generated by the Bagle.a, Bagle.f, Bagle.g, BubbleBoy, MyDoom.b, MyDoom.m, MyDoom.u, Netsky.d, and Sobig.f viruses [Martin, 2005]. Each virus infects a virtual machine and the emails it sends are recorded. The virtual machines are seeded with an actual user's address book so the viruses can exhibit realistic sending behaviors. Two viruses are particularly interesting: BubbleBoy does not need an attachment to propagate and uses Outlook rather than its own SMTP engine, and MyDoom.m uses highly polymorphic message bodies and subject lines.

4.3.2 Experimental procedure

Our primary experiment compares performance on the nine viruses in our dataset for varying numbers of training users and model topics. For each experiment, we select a training set of the appropriate number of users and a test set of one user. We choose the users uniformly at random without replacement, except that we require the test user to have sent at least 100 emails.[5] For each virus, we simulate an infection by injecting 100 emails from that virus' trace into the test user's email stream.

LDA and the mixture models classify generatively by choosing the best fit between normal and virus models, while the SVM is discriminative and produces a classifica-

[5]This excludes 40 of the 151 Enron employees in the dataset.

tion without modeling the classes themselves. The normal user models are trained on the randomly selected user training set and the virus models are trained on the traces from the eight other viruses; the SVM's training set includes both sets with appropriate labels. We use default settings for SVM-Light.

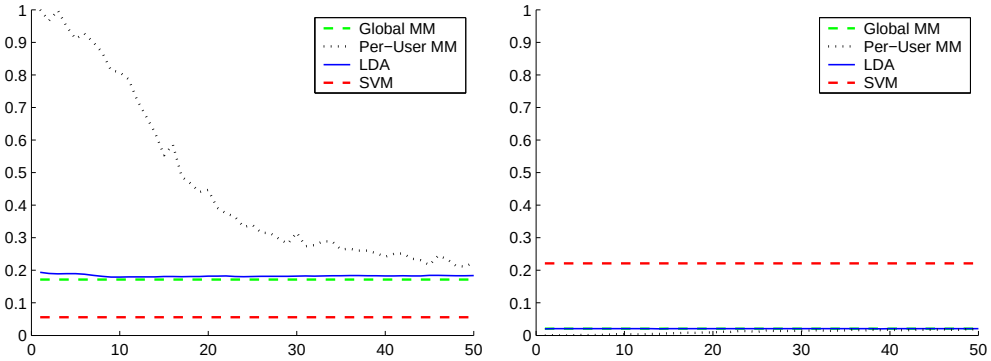
We train the models and then hold out the first 50 emails from the test user and (for LDA and the per-user mixture model) update the user-specific parameters based on up to 50 held-out emails. We then test on the 51st onward. This allows us to measure the performance of the algorithms against the number of emails seen from a new user. Each setting is run five times with different training and test sets for each number of held out emails from 1 to 50.[6]

We assume that our system has access to every email sent from a network of end-user machines and is able to accurately determine which user or machine sent each message. Some viruses attempt to bypass an organization's outgoing email servers (e.g., by including their own SMTP engines), however, transparent SMTP redirection or stateful packet inspection by a firewall can be used to enforce our assumption.

4.3.3 Results

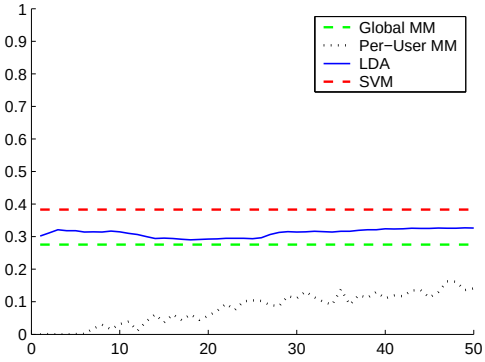
We show graphs of the learners' performance on five viruses in Figure 4.2 and final numbers for all nine viruses in Table 4.3. In this setting, a false positive (FP) is a normal email misclassified as a virus and a false negative (FN) is a virus email

[6]We train on the most recent emails in the held-out set to avoid the introduction of gaps in a user's stream of messages.

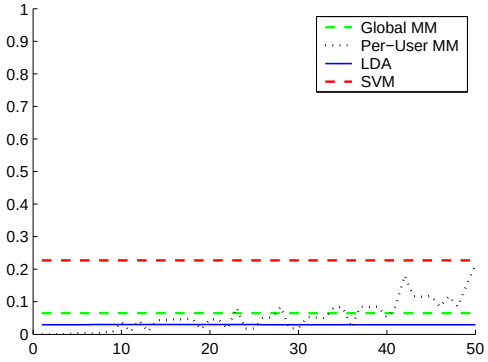


(a) Bagle.a false positives.

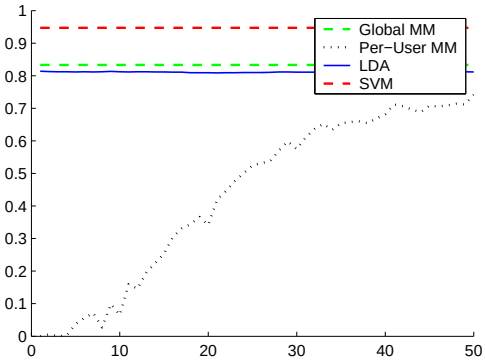
(b) Bagle.a false negatives.



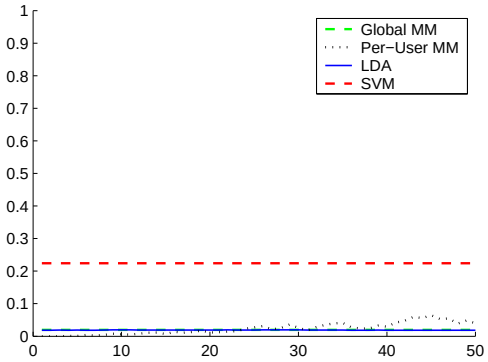
(c) Bagle.f false negatives.



(d) BubbleBoy false negatives.



(e) MyDoom.m false negatives.



(f) Sobig.f false negatives.

Figure 4.2: In these graphs, the horizontal axis gives the number of training emails seen from the test user. False positive rates (normal emails classified as virus) for all algorithms are nearly identical for the nine viruses; we show Bagle.a in (a). The false negatives (virus emails classified as non-virus) show more interesting behavior. We show graphs for five different viruses in (b) through (f).

misclassified as normal. The FP graphs for all nine viruses show very similar trends and provide little information that is not available in Table 4.3, so we only show one FP graph. Behavior on FNs is more varied, and we show the five most interesting viruses. Bagle.g, MyDoom.u, and Netsky.d have FN graphs nearly identical to Bagle.a. MyDoom.b has a graph similar to Bagle.f, but for MyDoom.b, LDA and GMM do 6–7% better and the SVM does much worse at 89% FNs.

The graphs in Figure 4.2 are averaged over five runs each of three and ten topics for LDA, the GMM, and the PMM. LDA and the GMM both have around 7% fewer FPs but 4% more FNs with ten topics, and the PMM has 10% fewer FPs and 5% more FNs with ten topics.

The SVM consistently has the lowest false positive rate. Although LDA generally improves by a few percent with more per-user data, the FP rates of GMM and LDA are nearly identical. The interesting part of Figure 4.2a is the PMM curve: it begins with a near-100% FP rate as it overfits to the first few emails (classifying everything as a virus), but with more data it improves and approaches the GMM and LDA.

The false negative rates are more interesting, as we see varying performance across viruses. The SVM has the *highest* FN rates in all graphs, while the GMM and LDA again have similar performance. The PMM starts off with very low FNs due to overfitting the user model, but as it generalizes it starts to misclassify virus emails.

The FN graph for MyDoom.m stands out, with terrible performance by all learners. MyDoom.m is a polymorphic virus that exhibits large variation in both subject

	False Negatives				False Positives			
	LDA	GMM	PMM	SVM	LDA	GMM	PMM	SVM
Bagle.a	2%	2%	2%	22%	18%	17%	22%	6%
Bagle.f	33%	28%	14%	38%	16%	15%	18%	5%
Bagle.g	2%	2%	3%	23%	20%	16%	18%	5%
BubbleBoy	3%	7%	21%	23%	16%	15%	21%	5%
MyDoom.b	26%	22%	9%	89%	14%	12%	12%	3%
MyDoom.m	81%	83%	74%	95%	11%	10%	11%	3%
MyDoom.u	3%	3%	2%	22%	16%	15%	23%	5%
Netsky.d	2%	2%	2%	22%	17%	16%	22%	5%
Sobig.f	2%	2%	4%	22%	18%	17%	22%	5%

Table 4.3: False positives/false negatives.

line and body, which makes it difficult to classify. Again the PMM initially classifies everything as a virus, but its FNs increase quickly. No learner correctly classifies more than 26% of MyDoom.m emails in the end.

Table 4.3 shows the FP and FN rates for our experiments. While we see high variation in FN rates across viruses, the FP rates remain relatively constant. This is not surprising: between any two experiments, the only difference relevant to a FP is a substitution of one of eight training viruses. FNs, however, show more variability because the email being classified is different.

Three related observations on our results merit investigation: the PMM does not tend to perform as well as the GMM even after training on 50 user emails, LDA does not show significant improvement over the 50 emails, and the GMM performs almost equivalently to LDA. We expect to see better per-user performance for the PMM and possibly LDA with more hold-out emails, since the PMM clearly improves up to the 50th email. Dataset limitations, however, preclude using more. It is also

possible that our feature set does not encode information that would allow the per-user models to gain significant advantage over global models. We conjecture that a more comprehensive feature set and feature selection algorithm, and perhaps increased numbers of hold-out emails, would allow LDA to consistently outperform the GMM.

4.4 Summary

In this chapter, we present an LDA model that combines global and per-user components for email virus detection. Our experimental results show that this system immediately provides acceptable performance for new users even with our conservative feature set and in the long run remains competitive with more specific per-user models. These results also show that, despite room for improvement in per-user specialization, LDA performs competitively with a simple support vector machine.

The combination of per-user and global models has the potential to react quickly to global changes while providing superior long-term performance. This work describes an approach that shows promise for increasing the ability of machine learning systems to defend users from novel viruses.

Chapter 5

Conclusion

5.1 Open problems

The study of attacks on machine learning systems is a relatively new field of inquiry. This thesis has provided a foundation for analyzing attacks and building learning systems that are robust to them, but this work is only the beginning. Here we outline promising directions for future research in this field.

5.1.1 Building defenses

In this thesis, we have presented our framework as a first step to making learning secure against attack. We argue that categorizing and analyzing attacks within our framework draws out commonalities that suggest general defenses. In Section 2.4 we go through different areas of the taxonomy and examine potential defenses for each

one. In Chapter 4 we investigate a learning algorithm that combines local and global information to combat attacks on multiple levels.

A general framework for measuring the adversarial effort required to perform an attack as well as the effectiveness of the defense would be of great value in designing secure learning systems. The ACRE-learning framework of Lowd and Meek [2005b] provides a computational analysis of the complexity of reverse engineering hypotheses from certain classes using probing in an *Exploratory* attack. Both robust statistics and online prediction with experts offer certain theoretical and quantitative tools that might be useful to build a more general framework for evaluating and constructing defenses across the space of attacks, as well as engineering specific defenses for attacks against particular learning systems.

5.1.2 Information

Several questions about information are crucial to understanding the interactions between attacker and defender for constructing secure learning systems.

Hiding information to prevent reverse engineering

We argue in Section 2.4 that the defender may be able to gain some advantage over the attacker in reverse engineering attacks if certain information is hidden from the attacker, such as the training data, feature set, and perhaps even hypothesis space. It is almost certainly true that lack of such information would hinder an attacker to

some degree, but this suggestion calls to mind questionable *security through obscurity* strategies. But there is another paradigm for security resting in the secrecy of information: hiding a cryptographic secret key fully protects the security of encrypted information in a good cryptosystem. The question we must answer is whether secure learning has the property that hiding certain information will make the attacker's task prohibitively difficult, as in the case of the cryptographic key. Rivest [1991] has noted the similarity between the learning problem and the cryptanalysis problem, and our question is similar but not exactly the same: we want to equate the cryptanalysis problem with the attacker's task of reverse engineering the learner's hypothesis to find false negatives.

Measuring the value of information

Measuring the value of information would be a useful tool for many purposes in building secure learning systems. There are several potential sources of quantitative measurements of information value, for example:

1. In the focused attack against SpamBayes, we show that as the attacker knows more of the target email, the probability of success goes up (Figure 3.4).

2. In the LDA experiments, we explore how the performance of learners improve as they have more instances to train on.

3. Lowd and Meek [2005b] formalize reverse engineering as the ACRE-learning

problem, a quantification of the complexity of reverse engineering classifiers in terms of queries.

A means of incorporating one or more of these sources into a quantitative analysis of security, such as considering them in the cost function, would be useful for security analysis.

Secrecy as a primary concern

We observe that an attacker may try to gain information about the internal state of a machine learning system not only to more effectively reverse engineer the system, but also for the sake of the information itself. For example, sensitive medical information may be encoded in the internal state of a hospital's email spam filter that has trained on emails from medical staff. The question of information secrecy may be a matter not merely of attack resistance but also of personal information security, business competitive secrecy, or even regulatory compliance.

Hiding attacks

So far we have discussed the attacker's information about the defender, but another consideration is the defender's information about the attacker. The attacker may have at least two good reasons for wanting to hide attack information from the defender. First, the attacker may want to prevent the defender from discovering that an attack is in progress. The attack may need time to be effective, or once it is

effective the attacker may gain benefit in proportion to the time before the defender notices and stops the attack. Second, the attacker may want to leave few forensic traces that might point to the source of the attack, either to prevent apprehension by law enforcement or to create uncertainty about where to look for the next attack.

One natural way to account for these considerations is to incorporate the divulgence of attacker information into the attacker's cost function; however, representing the cost of such information accurately may be difficult.

5.1.3 Specificity and scope of information

We have explored the use of global (cross-user) information versus local (user-specific) information in the domain of email virus detection in Chapter 4. A natural question is whether local information is more useful in combatting *Targeted* attacks and global information is more useful against *Indiscriminate* attacks. If so, then systems that successfully combine the two, such as the LDA system we present, will be especially useful.

This question is important, for example, in an organization-wide installation of a spam filter such as SpamBayes. One choice for training is to train a global model on email messages received by all users in the organization. Another choice is to train separate models for each user. The latter may be able to learn more precisely what sort of email each user normally receives, but the former can take advantage of more accumulated data from all the users. A *Targeted* attack aimed at one user

may be difficult to prevent without specific familiarity with that user's normal email distribution; on the other hand, an *Indiscriminate* attack will not be shaped for any particular user and the additional data from all users is likely to be useful. At this point, such conclusions do not have conclusive support, but it would be useful to determine whether there is significant advantage to global information, local information, or a combination of the two, and how closely this correlates with the **Specificity** of an attack.

5.2 Review of contributions

In this thesis, we have developed a framework for describing and analyzing attacks against machine learning systems, we have demonstrated effective attacks against the SpamBayes machine learning system, and we have proposed several defense ideas and successfully implemented one defense in the context of email virus detection.

In Chapter 2, we present the framework, which comprises a three-axis taxonomy and a cost-sensitive game. We survey attacks against machine learning systems in the literature and categorize them within our taxonomy. The taxonomy brings out interesting similarities among the attacks in the literature, as well as highlighting areas of the attack space that have received less attention. Furthermore, the taxonomy's division of the attack space into classes leads naturally to developing defenses for each class. We present ideas for defenses appropriate to each class of attack; we find that the taxonomy organizes attacks into natural classes for proposing general defenses.

In Chapter 3, we demonstrate effective attacks against the SpamBayes statistical machine learning spam filter. We design and implement two *Causative Availability* attacks, motivated in part by the taxonomy. With these two attacks we explore the **Specificity** spectrum: one is *Targeted* and one is *Indiscriminate*. We show that an adversary can effectively disable the SpamBayes spam filter with relatively little system state information and relatively limited control over training data. Our Usenet dictionary attack causes misclassification of 36% of ham messages with control over only 1% of the training messages, rendering SpamBayes unusable. Our focused attack changes the classification of the target message 60% of the time when it has only a 30% chance of guessing each of the target's tokens. These experiments demonstrate that SpamBayes is vulnerable to learning-based attacks, as well as providing a clear example of how the taxonomy can aid the analysis of attacks.

In Chapter 4, we turn to *Exploratory Integrity* attacks in the form of email viruses. The viruses use polymorphic engines for generating email text and other evasive tactics designed to defeat classifiers. We present a learning technique, Latent Dirichlet Allocation (LDA), to combat these attacks by combining local and global information about user behavior. By combining information at different scales, LDA and similar techniques have the potential to resist attacks across the *Targeted*/*Indiscriminate* spectrum.

Our framework describes structure in the space of attacks on machine learning systems and presents it in a format useful for analysis. The SpamBayes work uses the

framework in the creation of attacks and shows that such attacks can be successful. The LDA system and the analysis of defenses within the taxonomy build defensive techniques using our framework as a base. This thesis shows that the space of attacks against machine learning systems has a structure that facilitates analysis and provides a foundation for building secure learning systems.

Notation

Symbol	Description	Pages
$\mathbf{a}$	Attack data instance in $\mathcal{X}$	50, 87
A_E	Procedure for selecting evaluation distribution: $\mathcal{X}^N \times \Omega \mapsto \mathfrak{D}$	20–22, 36
A_T	Procedure for selecting training distribution: $\mathcal{X}^N \times \Omega \mapsto \mathfrak{D}$	22, 36
α	Prior over topic distributions	68, 69, 71
b	Prior belief for token scores	45
β	Feature distribution parameters ($T \times F$ matrix)	68, 69, 71
C	Cost function: $\mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$	9, 21, 22, 37

Symbol	Description	Pages
χ^2_{2n}	Chi-square distribution function with $2n$ degrees of freedom	46
D	Procedure for selecting hypothesis: $(\mathcal{X} \times \mathcal{Y})^N \mapsto \Omega$	9, 20–22, 26, 47, 48
$\mathfrak{D}$	Space of distributions over $(\mathcal{X} \times \mathcal{Y})$	87, 89
$1 - \delta$	Probability of learning ϵ-good hypothesis (PAC-learning)	25, 99
E	Number of emails	47, 48, 54, 58, 68, 69, 74
$\mathbf{E}$	Evaluation set in $(\mathcal{X} \times \mathcal{Y})^N$	9, 21–23, 26, 47, 48
ϵ	Probability of error (PAC-learning)	25, 87, 99
F	Number of features	68, 87
f	Hypothesis (classifier): $\mathcal{X} \mapsto \mathcal{Y}$	9, 10, 21–23, 34, 36, 47, 48, 87
f^*	Best hypothesis: $\mathcal{X} \mapsto \mathcal{Y}$	9, 10

Symbol	Description	Pages
g	Expert prediction: $g_m^{(k)}$ is prediction of m^{th} expert in k^{th} round	37, 87
H	Ham score of an email	46
h	Hypothesis combining experts: $\mathcal{Y}^M \mapsto \mathcal{Y}$	37, 38
I	Message score: $I(\mathbf{x})$ is score of $\mathbf{x}$	46, 50, 87
$I_{\mathbf{a}}$	Message score after adding $\mathbf{a}$ to training	50
K	Number of repetitions of a game	37, 38, 87
k	Index for game steps (up to K)	37, 38, 87
ℓ	Loss function: $\mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^{0+}$	9, 10
λ	Trade-off between loss and regularization terms	10
M	Number of experts	36, 37, 87

Symbol	Description	Pages
m	Index for experts (up to M)	37, 87
N	Number of data points	9, 87, 89
N_H	Number of *ham* emails	45
$N_H(w)$	Number of *ham* emails in which token w appears	45
N_S	Number of *spam* emails	45
$N_S(w)$	Number of *spam* emails in which token w appears	45
$N(w)$	Number of emails in which token w appears	45
$\nu(\mathbf{x})$	Set of tokens SpamBayes uses from message $\mathbf{x}$	46
Ω	Space of hypotheses $f : \mathcal{X} \mapsto \mathcal{Y}$	9, 10, 87
$\mathbb{P}$	Evaluation and retraining distribution in $\mathfrak{D}$	20–22, 37, 87, 89

Symbol	Description	Pages
$\mathbf{p}$	Message distribution (vector of token probabilities)	50, 51
$\mathbb{P}_{\mathrm{E}}$	Evaluation distribution in $\mathfrak{D}$	21, 22
p_i	Probability of the i^{th} token appearing in a message	51
$P_R(w)$	Raw token score of w	45
$P_S(u)$	Smoothed token score of u	50
$P_S(w)$	Smoothed token score of w	45, 46, 50
$\mathbb{P}_{\mathrm{T}}$	Training distribution in $\mathfrak{D}$	20–22
p_w	Attacker's probability of guessing each word in target message	54, 58
ϕ	Distribution over topics	47, 48, 54, 58, 68, 69, 71, 74
$\mathbb{R}$	Space of real numbers	87, 89

Symbol	Description	Pages
$\mathbb{R}^{0+}$	Space of non-negative real numbers	87
ρ	Regularization function: $\Omega \mapsto \mathbb{R}$	9, 10
S	Spam score of an email	46
s	Strength of prior	45
T	Number of topics	68, 71, 87, 89
$\mathbf{t}$	Vector of T email counts	71
θ_0	SpamBayes *ham*/*unsure* threshold for determining message label	46
θ_1	SpamBayes *spam*/*unsure* threshold for determining message label	46
U	Number of users	47, 48, 54, 58, 68, 69, 74
u	A second token (word)	50, 89

Symbol	Description	Pages
w	A token (word)	45, 46, 50, 87, 89
$\mathbf{X}$	Training set in $(\mathcal{X} \times \mathcal{Y})^N$	9, 10, 21, 22, 26, 47, 48
$\mathbf{x}$	Data instance in $\mathcal{X}$	9, 10, 21, 22, 34, 37, 46–48, 50, 51, 54, 58, 68, 69, 71, 74, 87
$\mathcal{X}$	Space of data instances	9, 34, 87, 89
y	Data label in $\mathcal{Y}$	9, 10, 21, 22, 37, 91
$\hat{y}$	Predicted data label	37
$\mathcal{Y}$	Space of data labels; for classification $\mathcal{Y} = \{0, 1\}$	9, 37, 87, 89, 91
z	Topic	68, 69, 71

Glossary

attack email

An email sent by the attacker as part of an attack

attacker

The person or agent whose goal is to compromise the assets of a system; also, the second player in the adversarial game

***Availability* attack**

Availability attacks cause denial of service, usually via false positives

***Causative* attack**

Causative attacks influence learning with control over training data

classifier

A function that gives a class label to each instance

contamination assumption

The assumption that an attacker can construct data that the learner will use for training

cost function

A function that assigns a value to each correct or incorrect labeling for any given instance; a negative cost is a benefit

cross-validation

A technique in which a dataset is split into several pieces, each of which serves independently as training and validation data in different runs

defender

The person or agent whose goal is to protect a system from attacks; also, the first player in the adversarial game

dictionary attack

An attack in which the attacker sends emails containing entire dictionaries, making a spam filter unusable when it trains on them

empirical risk

The total empirical loss computed over a dataset, used as an approximation of expected loss on the data distribution

evaluation set

Data on which the classifier's performance is evaluated

evaluation step

A step in which the current hypothesis is applied to evaluation data and the parties involved incur cost based on the results

expert

A classifier we use in a combined prediction method but about which we make no assumptions

***Exploratory* attack**

Exploratory attacks exploit misclassifications but do not affect training

empirical risk minimization

A procedure in which the learner minimizes an objective that combines an empirical risk term and a regularization term

false negative (FN)

A malicious instance misclassified as a negative

false positive (FP)

A benign instance misclassified as a positive

feature deletion

When features present in the training data, and perhaps highly predictive of an instance's class, are removed from the evaluation data by the attacker.

focused attack

An attack in which the attacker sends emails containing tokens known to appear in a particular target email, causing that email to be blocked by the spam filter

global information

Information collected across many users or otherwise encompassing a broad scope

global mixture model

A model of events drawn from topics where there is no distinction among users

ham

Good, legitimate email

hypothesis

A function selected by a learner; in our setting, a classifier

***Indiscriminate* attack**

Indiscriminate attacks encompass a wide class of instances

instance

One unit of data; a data point

***Integrity* attack**

Integrity attacks compromise assets via false negatives

iterated game

A game in which the sequence of moves is repeated over many iterations

LDA

Latent Dirichlet Allocation, a generative graphical model that represents events as being generated from one of several topic distributions and automatically infers topics

local information

Information collected specifically from one user or otherwise focusing on a narrow scope

loss function

A function that penalizes errors; the criterion for judging classifier performance

message score

A score between 0 and 1 with low scores indicating *ham* and high scores indicating *spam*

MIME

Multipurpose Internet Mail Extensions, a standard for supporting multi-part message bodies, including non-text parts, in email messages

move

One play in the adversarial game, in which a player chooses a strategy

n-gram

A feature formed by taking n consecutive items, such as words or bytes

negative class

The benign class of data

one-shot game

A game in which each move occurs once

overfitting

Selecting a hypothesis that is over-specialized for the training data (a look-up table for the training data is an extreme case of overfitting)

PAC-learning

Probably Approximately Correct learning–a learning formalism in which the goal is to learn, with probability at least $1 - \delta$, a hypothesis with probability at most ϵ of making an incorrect prediction

per-user mixture model

A model of events drawn from topics where each user has their own parameters and users are not connected in any way

positive class

The malicious class of data

probing

Querying a classifier with instances and using the responses in an attack

procedure

A method, such as a learning algorithm, for selecting hypotheses given a dataset

regret

The difference between the loss of a composite learner and the loss of the best overall expert in hindsight

regularization

A means of penalizing hypothesis complexity, used primarily to prevent overfitting

robust statistics

A set of techniques that limit the impact of a small fraction of adversarial training data deviating from a known model

scanning

Randomly or systematically making connections across IP address space, often used by worms to find new victim hosts

secure learning

Learning that succeeds in adversarial conditions

security goal

A requirement that, if violated, results in the partial or total compromise of an asset

security through obscurity

The idea that keeping details of a system secret will keep it safe from attack, usually eschewed by security experts

spam

Bad, unsolicited email

spam score

Synonym for **message score** or **token score**, depending on context

SpamBayes

A popular machine learning spam filter that assigns *spam*, *ham*, and *unsure* labels to emails based on the words they contain

stationarity

The assumption that training data and evaluation data are drawn from the same distribution, or in general that the data distribution does not change over time

step

Synonym for **move**

supervised classification

The learning setting in which a learner finds a hypothesis to classify instances, training on data instances with class labels

support vector machine (SVM)

A classifier that learns to maximize the margin, or separation between the classes, and is often highly effective

***Targeted* attack**

Targeted attacks focus on a particular instance

threat model

A profile of attackers, describing motivation and capabilities

token score

Smoothed estimate of how much a token indicates that a message is spam

topic

A collection of events (emails) drawn from the same distribution over features

training set

A dataset used for training

training step

A step in which a learning procedure is applied to training data to produce a new hypothesis

TREC corpus

Email corpus based on the Enron email dataset, constructed for the 2005 Text Retrieval Conference spam filtering competition

unsure

Label indicating filter is uncertain about classification

Usenet dictionary

A dictionary constructed from common tokens in a Usenet corpus rather than a plain English dictionary

zero-sum game

A two-player game in which costs for the players add up to zero (or any other fixed value) for each possible outcome

Index

Bibliography

Marco Barreno, Blaine Nelson, Russell Sears, and Anthony D. Joseph. User model transfer for email virus detection. In *Proceedings of the First Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, June 2006a.

Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the ACM Symposium on InformAtion, Computer, and Communications Security (ASIACCS'06)*, pages 16–25, March 2006b.

Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. Technical Report UCB/EECS-2008-43, EECS Department, University of California, Berkeley, April 2008. `http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-43.html`.

Peter J. Bickel and Kjell A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics*, volume 1. Prentice-Hall, Inc., 2nd edition, 2001.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, 2003. ISSN 1533-7928.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Andreas Christmann and Ingo Steinwart. On robustness properties of convex risk minimization methods for pattern recognition. *Journal of Machine Learning Research (JMLR)*, 5:1007–1034, 2004. ISSN 1533-7928.

Simon P. Chung and Aloysius K. Mok. Allergy attack against automatic signature generation. In *Recent Advances in Intrusion Detection (RAID)*, pages 61–80, 2006.

Simon P. Chung and Aloysius K. Mok. Advanced allergy attacks: Does a corpus really help? In *Recent Advances in Intrusion Detection (RAID)*, pages 236–255, 2007.

William W. Cohen. Enron email dataset. `http://www.cs.cmu.edu/~enron/`.

Gordon Cormack and Thomas Lynam. Spam corpus creation for TREC. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, July 2005.

Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108, Seattle, WA, 2004. ACM Press.

Mark Dredze, Reuven Gevaryahu, and Ari Elias-Bachrach. Learning fast classifiers for image spam. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, 2007.

Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Salvatore J. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In D. Barbara and S. Jajodia, editors, *Proceedings of the Workshop on Data Mining for Security Applications*. Kluwer, 2002.

Ronald A. Fisher. Question 14: Combining independent tests of significance. *American Statistician*, 2(5):30–30J, 1948.

Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: Formal reasoning and practical techniques. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 59–68, 2006.

Amir Globerson and Sam Roweis. Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 353–360, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: http://doi.acm.org/10.1145/1143844.1143889.

Paul Graham. A plan for spam. http://www.paulgraham.com/spam.html, August 2002.

Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel.

Robust Statistics: The Approach Based on Influence Functions. Probability and Mathematical Statistics. John Wiley and Sons, 1986.

Peter J. Huber. *Robust Statistics*. Probability and Mathematical Statistics. John Wiley and Sons, 1981.

Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*. MIT Press, 1999.

Michael I. Jordan. Graphical models. *Statistical Science*, 19(1):140–155, 2004.

Christoph Karlberger, Günther Bayler, Christopher Kruegel, and Engin Kirda. Exploiting redundancy in natural language to penetrate Bayesian spam filters. In *WOOT'07: Proceedings of the first conference on First USENIX Workshop On Offensive Technologies*, 2007.

Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.

Hyang-Ah Kim and Brad Karp. Autograph: Toward automated, distributed worm signature detection. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

Bryan Klimt and Yiming Yang. Introducing the Enron corpus. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, July 2004.

Aleksandar Lazarevic, Levent Ertöz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In Daniel Barbará and Chandrika Kamath, editors, *Proceedings of the Third SIAM International Conference on Data Mining*, May 2003.

Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, 2005a.

Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647, 2005b.

Ricardo A. Maronna, Douglas R. Martin, and Victor J. Yohai. *Robust Statistics: Theory and Methods*. John Wiley and Sons, New York, 2006.

Steve Martin, Anil Sewani, Blaine Nelson, Karl Chen, and Anthony Joseph. Analyzing behavioral features for email classification. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, 2005.

Steven L. Martin. Learning on email behavior to detect novel worm infections. Master's thesis, University of California, Berkeley, 2005.

Tony A. Meyer and Brendon Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, July 2004.

David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems*, 24(2):115–139, 2006. ISSN 0734-2071. doi: http://doi.acm.org/10.1145/1132026.1132027.

Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. D. Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. In *Proceedings of the First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, April 2008.

James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 226–241, May 2005.

James Newsome, Brad Karp, and Dawn Song. Paragraph: Thwarting signature learning by training maliciously. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*, September 2006.

Ronald L. Rivest. Cryptography and machine learning. In *Advances in Cryptology (ASIACRYPT'91)*, pages 427–439. Springer Verlag, 1991.

Gary Robinson. A statistical approach to the spam problem. *Linux Journal*, March 2003.

D. Sculley, Gabriel M. Wachman, and Carla E. Brodley. Spam filtering using in-

exact string matching in explicit feature space with on-line linear classifiers. In *Proceedings of the Fifteenth Text Retrieval Conference (TREC)*, 2006.

Larry J. Seltzer. Security watch: MyDoom reappears. *Security Watch Newsletter*, February 2005. http://www.pcmag.com/article2/0,1759,1767805,00.asp.

Cyrus Shaoul and Chris Westbury. A USENET corpus (2005-2007), October 2007. http://www.psych.ualberta.ca/~westburylab/downloads/usenetcorpus.download.html.

Jitesh Shetty and Jafar Adibi. The Enron email dataset: Database schema and brief statistical report. http://www.isi.edu/~adibi/Enron/Enron.htm.

Sophos Corporation. Top 10 viruses reported to Sophos in 2005. Online, December 2005. http://www.sophos.com/virusinfo/topten/200512summary.html.

Salvatore J. Stolfo, Shlomo Hershkop, Chia-Wei Hu, Wei-Jen Li, Olivier Nimeskern, and Ke Wang. Behavior-based modeling and its application to email analysis. *ACM Transactions on Internet Technology (TOIT)*, 6(2):187–221, May 2006.

Kymie M. C. Tan, Kevin S. Killourhy, and Roy A. Maxion. Undermining an anomaly-based intrusion detection system using common exploits. In *Recent Advances in Intrusion Detection (RAID)*, pages 54–73, 2002.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11): 1134–1142, November 1984.

Leslie G. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 560–566, August 1985.

David Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 78–87, New York, NY, USA, October 2004. ACM Press.

Ke Wang, Janak J. Parekh, and Salvatore J. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Recent Advances in Intrusion Detection (RAID)*, pages 226–248, 2006.

Zhe Wang, William Josephson, Qin Lv, Moses Charikar, and Kai Li. Filtering image spam with near-duplicate detection. In *Proceedings of the Fourth Conference on Email and Anti-Spam (CEAS)*, 2007.

Gregory L. Wittel and S. Felix Wu. On attacking statistical spam filters. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.